Universidade Federal de Minas Gerais
Escola de Engenharia da UFMG

PROJETO FINAL
Caça-Níqueis

Integrantes:
Italo Lelis de Carvalho
Thiago Rinco da Silveira

Professor:
Luciano Cunha de Araujo Pimenta

Curso: Engenharia de Controle e Automação
Disciplina: Sistemas Digitais (Prática)
Turma: PA1
Data: 28/06/2017

Descrição

O projeto consiste em um processador customizado com uma função definida, onde controlador e caminho de dados trabalham juntos em prol de um objetivo. O objetivo, neste caso, é um sistema que funcione como uma máquina caça-níquel, muito comum nos cassinos ao redor do mundo.

O dispositivo irá sortear 3 números e, caso esses três números sejam iguais, será indicado que o usuário ganhou a partida.

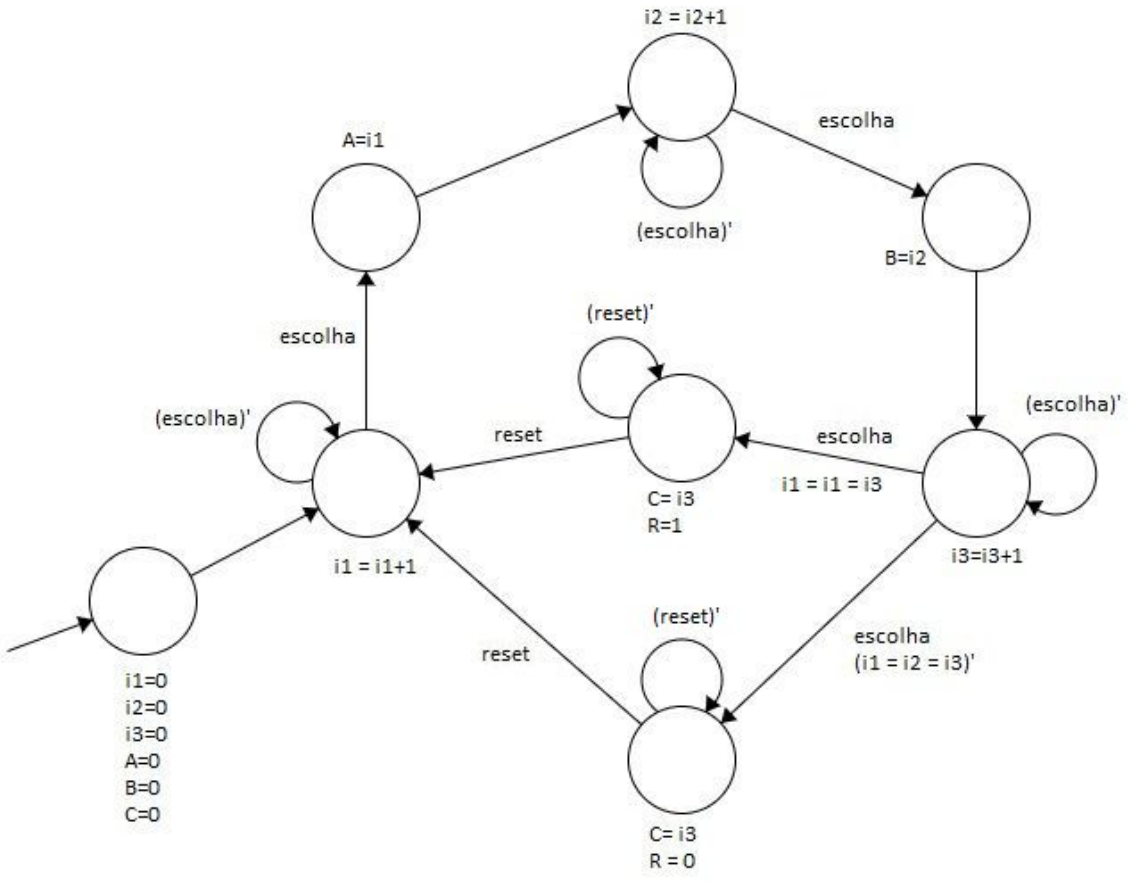
Para o sorteio dos números, será utilizado um contador ascendente. O contador irá somar 1 durante a subida do clock e, devido à velocidade do clock, não será possível prever o número que será salvo no registrador ao apertar o botão de selecionar. Isso fornece um certo nível de randomicidade ao projeto.

O controlador conta com uma máquina de estados, que administra o caminho de dados. Este utiliza três registradores para gravar os 3 números, um registrador para o resultado, um contador ascendente, que conte de 0 a 9, dois comparadores de igualdade e uma porta lógica ‘and’.

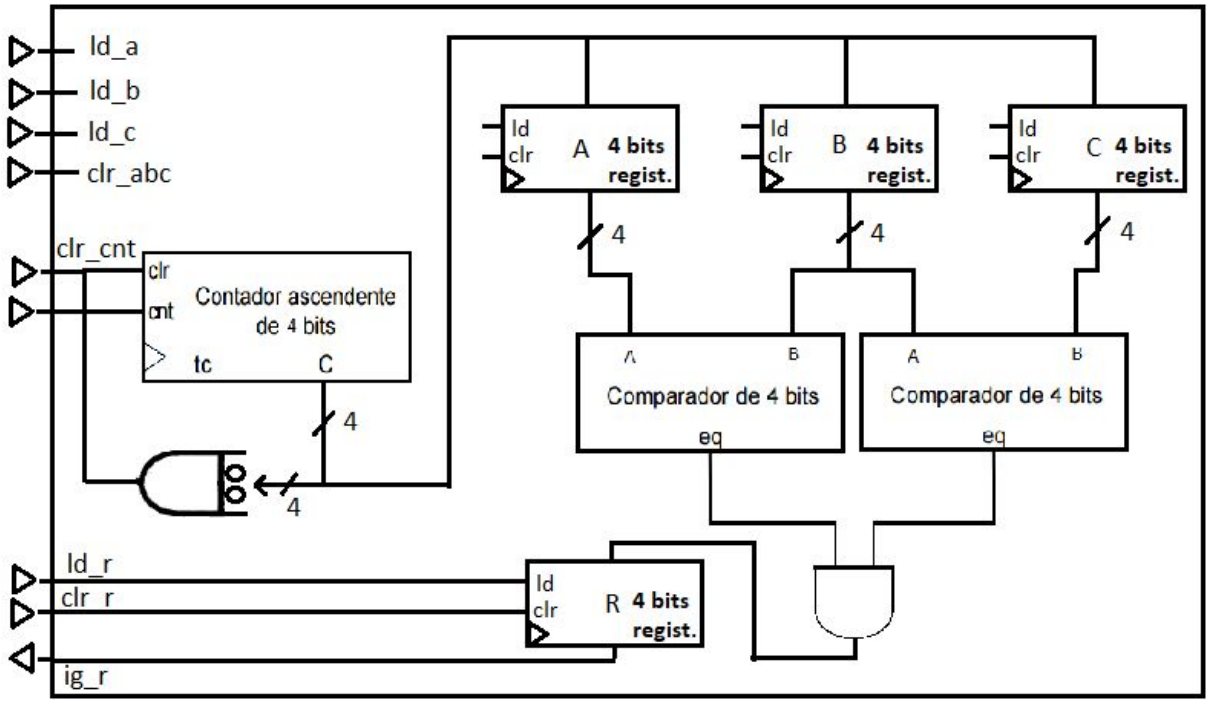
Projeto RTL

Passo 1: FSM de Alto Nível

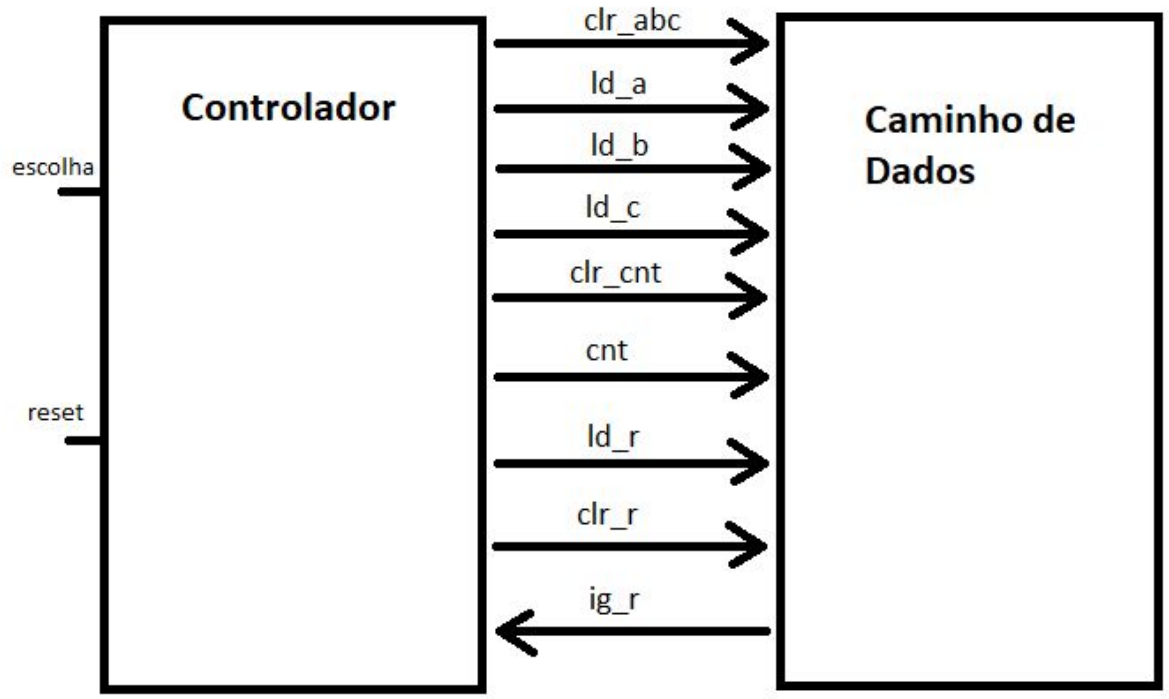
- Entradas: escolha (1 bit), reset (1 bit);
- Saídas: R(1 bit);
- Registradores: A,B,C (4 bits)



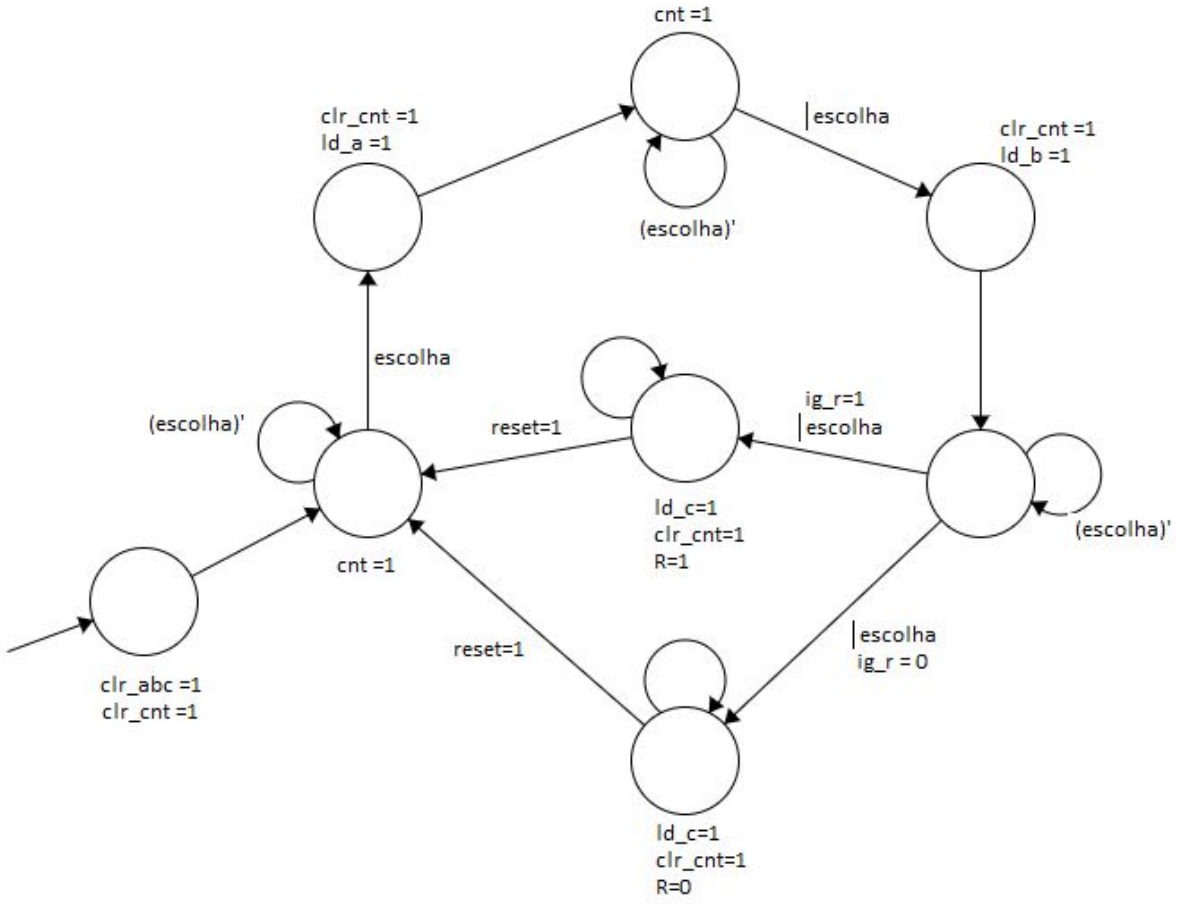
Passo 2: Caminho de Dados



Passo 3: Conexão entre caminho de dados e controlador



Passo 4:


cnt =1
clr_cnt =1
ld_a =1
escolha
clr_cnt =1
ld_b =1
(escolha)'
escolha
(escolha)'
reset=1
ig_r=1
escolha
ld_c=1
clr_cnt=1
R=1
(escolha)'
cnt =1
clr_abc =1
clr_cnt =1
reset=1
escolha
ig_r = 0
ld_c=1
clr_cnt=1
R=0

Código-Fonte

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

entity cassino is
   generic(
      constant min : integer := 0;
      constant max : integer := 9
   );
   port(
      clk      : in  std_logic;
      input    : in  std_logic;
      start    : in  std_logic;
      reset    : in  std_logic;
      output   : out std_logic_vector(1 downto 0);

      -- 7 segments display
      A: out std_logic_vector(6 downto 0);
      B: out std_logic_vector(6 downto 0);
      C: out std_logic_vector(6 downto 0);
      D: out std_logic_vector(6 downto 0)
   );

end entity;

architecture rtl of cassino is

   -- Build an enumerated type for the state machine
   type state_type is (s0, s1, s2, s3, compare, win, lose,
s1p, s2p, s3p, clear);

   -- Register to hold the current state
   signal state   : state_type;

   -- Create variables to store random numbers
   shared variable i1 : integer range 0 to max;
   shared variable i2 : integer range 0 to max;
   shared variable i3 : integer range 0 to max;

begin

   -- Logic to advance to the next state
   process (input, clk, start, reset)
   begin
      if reset = '0' then
         state <= s0;
```

```
   elsif (rising_edge(clk)) then
      case state is

      ---s0---
         when s0=>
            --
            i1 := min;
            i2 := min;
            i3 := min;
            output <= "00";
            A <= "0111111";
            B <= "0111111";
            C <= "0111111";
            D <= "1111111";
            --change condition
            if start = '0' then
               state <= s1;
            else
               state <= s0;
            end if;
      ---s1---
         when s1=>
            if    (i1 = 0) then C <= "0111111";
            elsif (i1 = 1) then C <= "0011111";
            elsif (i1 = 2) then C <= "0101111";
            elsif (i1 = 3) then C <= "0110111";
            elsif (i1 = 4) then C <= "0111011";
            elsif (i1 = 5) then C <= "0111101";
            elsif (i1 = 6) then C <= "0111110";
            elsif (i1 = 7) then C <= "0111111";
            elsif (i1 = 8) then C <= "0101111";
            elsif (i1 = 9) then C <= "0110111";
            else C <= "0110110";
            end if;
            if i1 = max then
               i1 := min;
            else
               i1 := i1 + 1;
            end if;
            --change condition
            if input = '0' then
               state <= s1p;
            else
               state <= s1;
            end if;
      ---s2---
         when s2=>
            if    (i2 = 0) then B <= "0111111";
```

```
            elsif (i2 = 1) then B <= "0011111";
            elsif (i2 = 2) then B <= "0101111";
            elsif (i2 = 3) then B <= "0110111";
            elsif (i2 = 4) then B <= "0111011";
            elsif (i2 = 5) then B <= "0111101";
            elsif (i2 = 6) then B <= "0111110";
            elsif (i2 = 7) then B <= "0111111";
            elsif (i2 = 8) then B <= "0101111";
            elsif (i2 = 9) then B <= "0110111";
            else B <= "0110110";
            end if;
            if i2 = max then
               i2 := min;
            else
               i2 := i2 + 1;
            end if;
            --change condition
            if input = '0' then
               state <= s2p;
            else
               state <= s2;
            end if;
      ---s3---
         when s3 =>
            if    (i3 = 0) then A <= "0111111";
            elsif (i3 = 1) then A <= "0011111";
            elsif (i3 = 2) then A <= "0101111";
            elsif (i3 = 3) then A <= "0110111";
            elsif (i3 = 4) then A <= "0111011";
            elsif (i3 = 5) then A <= "0111101";
            elsif (i3 = 6) then A <= "0111110";
            elsif (i3 = 7) then A <= "0111111";
            elsif (i3 = 8) then A <= "0101111";
            elsif (i3 = 9) then A <= "0110111";
            else A <= "0110110";
            end if;
            if i3 = max then
               i3 := min;
            else
               i3 := i3 + 1;
            end if;
            --change condition
            if input = '0' then
               state <= s3p;
            else
               state <= s3;
            end if;
```

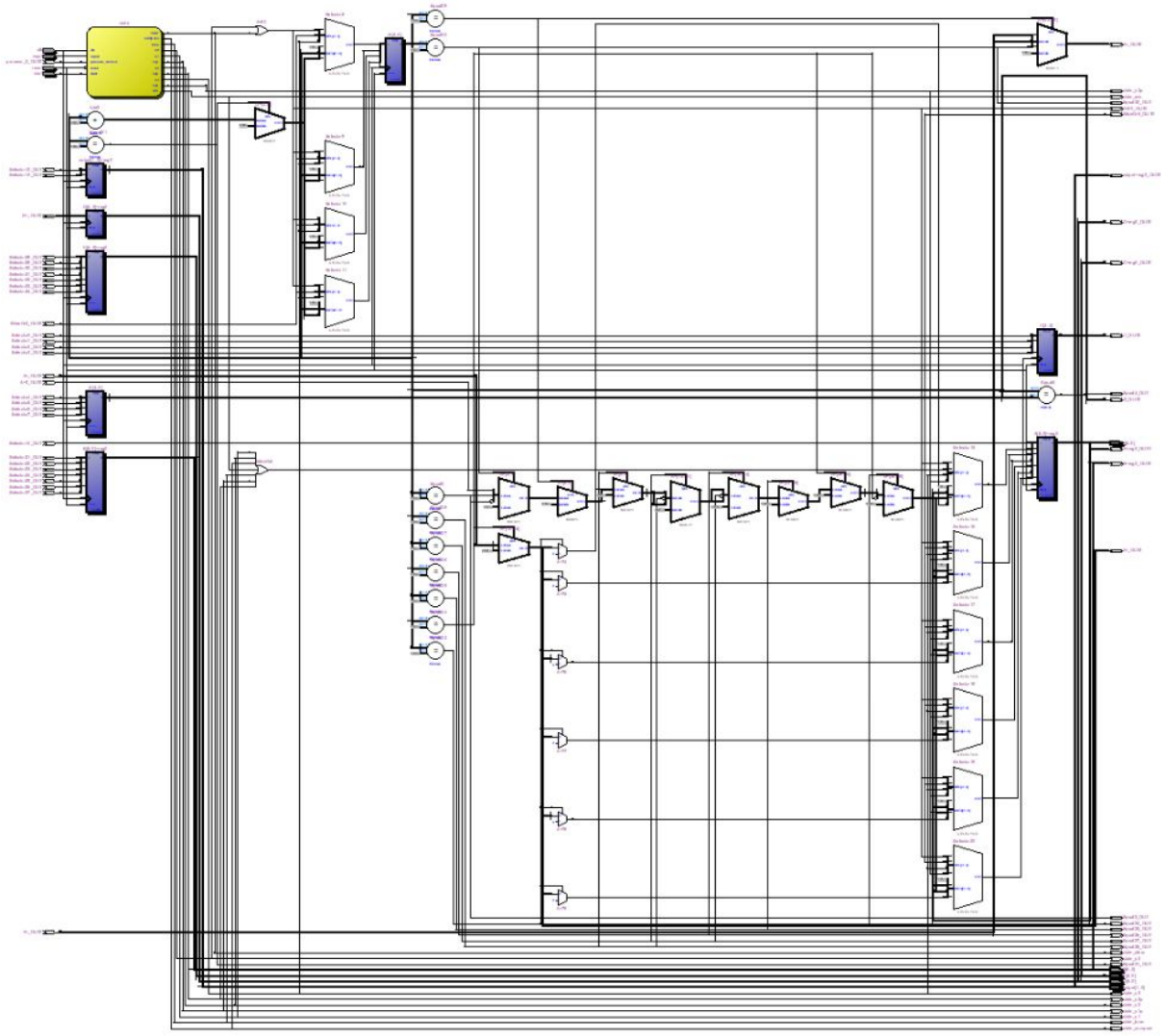
```
---compare---
   when compare =>
      --change condition
      if ((i1 = i2) and (i2 = i3)) then
         state <= win;
      else
         state <= lose;
      end if;
---win---
   when win =>
      output <= "11";
      --change condition
      if start = '0' then
         state <= clear;
      else
         state <= win;
      end if;
---lose---
   when lose =>
      output <= "00";
      --change condition
      if start = '0' then
         state <= clear;
      else
         state <= lose;
      end if;
---clear---
   when clear=>
      i1 := min;
      i2 := min;
      i3 := min;
      output <= "00";
      A <= "0111111";
      B <= "0111111";
      C <= "0111111";
      D <= "1111111";
      state <= s1;


---s1p---print(s1)---
   when s1p =>
      if    (i1 = 0) then C <= "1000000";
      elsif (i1 = 1) then C <= "1111001";
      elsif (i1 = 2) then C <= "0100100";
      elsif (i1 = 3) then C <= "0110000";
      elsif (i1 = 4) then C <= "0011001";
      elsif (i1 = 5) then C <= "0010010";
```

```
            elsif (i1 = 6) then C <= "0000010";
            elsif (i1 = 7) then C <= "1111000";
            elsif (i1 = 8) then C <= "0000000";
            elsif (i1 = 9) then C <= "0010000";
            else C <= "0000110";
            end if;
            --change condition
            if input = '1' then
               state <= s2;
            else
               state <= s1p;
            end if;
         ---s2p---print(s2)---
            when s2p =>
               if    (i2 = 0) then B <= "1000000";
               elsif (i2 = 1) then B <= "1111001";
               elsif (i2 = 2) then B <= "0100100";
               elsif (i2 = 3) then B <= "0110000";
               elsif (i2 = 4) then B <= "0011001";
               elsif (i2 = 5) then B <= "0010010";
               elsif (i2 = 6) then B <= "0000010";
               elsif (i2 = 7) then B <= "1111000";
               elsif (i2 = 8) then B <= "0000000";
               elsif (i2 = 9) then B <= "0010000";
               else B <= "0000110";
               end if;
               --change condition
               if input = '1' then
                  state <= s3;
               else
                  state <= s2p;
               end if;
         ---s3p---print(s3)---
            when s3p =>
               if    (i3 = 0) then A <= "1000000";
               elsif (i3 = 1) then A <= "1111001";
               elsif (i3 = 2) then A <= "0100100";
               elsif (i3 = 3) then A <= "0110000";
               elsif (i3 = 4) then A <= "0011001";
               elsif (i3 = 5) then A <= "0010010";
               elsif (i3 = 6) then A <= "0000010";
               elsif (i3 = 7) then A <= "1111000";
               elsif (i3 = 8) then A <= "0000000";
               elsif (i3 = 9) then A <= "0010000";
               else A <= "0000110";
               end if;
               --change condition
               if input = '1' then
```

```
                    state <= compare;
                else
                    state <= s3p;
                end if;
            end case;
        end if;
    end process;
end rtl;
```

Máquinas de Estado (Netlist Viewer)

RTL Viewer:



State Machine Viewer: