Relatório Prática 08 – 27/10/2023

Arthur Souza/João Paulo – PN1

Iniciou a prática criando uma entidade nomeado Datapath no software QUARTUS II 13.0, especificando qual chip está sendo utilizado, no caso o EP2C35F672C6.

A partir disso, adicionou-se arquivos de práticas anteriores que irão funcionar como componentes desse projeto, sendo eles o comparador(prática 4), o FlipFlopD(prática 5) e o Mean_4_clocks(prática 6). Além disso, para comunicação com a placa, foi adicionado um componente para tradução de palavras de 4 bits para display de 7 segmentos. A seguir o código do Datapath com os devidos componentes atualizados

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Datapath is
 port (
 Entrada : in std_logic_vector(3 downto 0);
 Fio_Load_E : in std_logic;
 Fio_Reset_MA : in std_logic;
 Fio_Descendo : in std_logic_vector(0 downto 0);
 Fio_Subindo : in std_logic_vector(0 downto 0);
 Fio_Atualizar : in std_logic;
 Fio_Clock : in std_logic;

 Fio_Maior : out std_logic;
 Fio_Igual : out std_logic;
 Fio_Menor : out std_logic;
 Subindo : out std_logic_vector(0 downto 0);
 Descendo : out std_logic_vector(0 downto 0);
 Media_DSP_7Seg : out std_logic_vector(6 downto 0)
 );
end Datapath;
architecture Arch of Datapath is
 component Reg_MA is
  generic (
   W : integer := 32
  );
  port (
   CLK : in std_logic;
   RESET : in std_logic;
   INPUT : in std_logic_vector(W - 1 downto 0);
   OUTPUT : out std_logic_vector(W - 1 downto 0)
  );
 end component;
 component Comparador is
  generic
  (
   DATA_WIDTH : natural := 16
  );
  port
  (
```

```
        a : in std_logic_vector ((DATA_WIDTH-1) downto 0);
        b : in std_logic_vector ((DATA_WIDTH-1) downto 0);
        maior : out std_logic;
        menor : out std_logic;
        igual : out std_logic
       );
      end component;
      component Bcd_7seg is
       port (
        entrada: in std_logic_vector (3 downto 0);
        saida: out std_logic_vector (6 downto 0)
       );
      end component;
      component RegW is
       generic(
        W: integer:=32
       );
       port(
        Load: in std_logic;
        clock: in std_logic;
        E: in std_logic_vector(W-1 downto 0);
        Q: out std_logic_vector(W-1 downto 0)
       );
      end component;
      signal Sai_a, Sai_b : std_logic_vector(3 downto 0);
      begin
      RegistradorE: RegW generic map (4) port map(Fio_Load_E, Fio_Clock, Entrada,
Sai_a);
      RegistradorMedia: Reg_MA generic map(4) port map(Fio_Clock, Fio_Reset_MA,
Sai_a, Sai_b);
      Comp: Comparador generic map(4) port map(Sai_a, Sai_b, Fio_Maior, Fio_Menor,
Fio_Igual);
      Led7segmentos: BCD_7seg port map(Sai_b, Media_DSP_7Seg);
      RegistradorS: RegW generic map (1) port map(Fio_Atualizar, Fio_Clock,
Fio_Subindo, Subindo);
      RegistradorD: RegW generic map (1) port map(Fio_Atualizar, Fio_Clock,
Fio_Descendo, Descendo);
      end Arch;
```

Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e Technology Map Viewer. Segue abaixo as imagens:

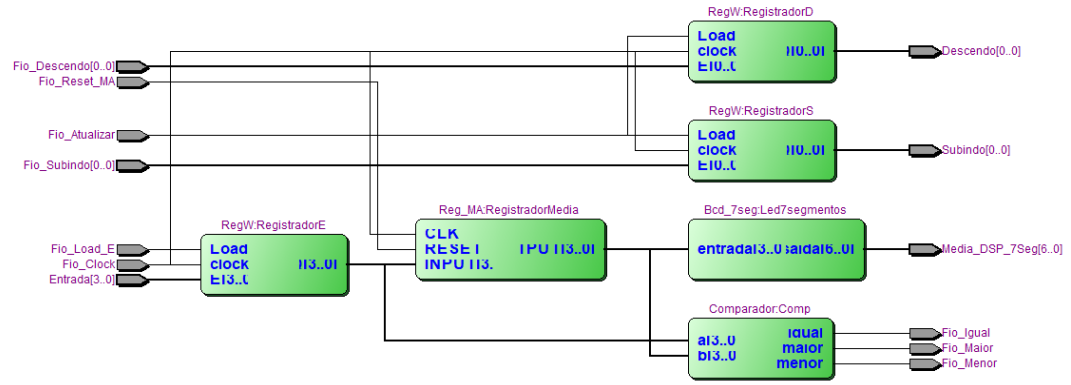

Figura 1:Circuito Datapath

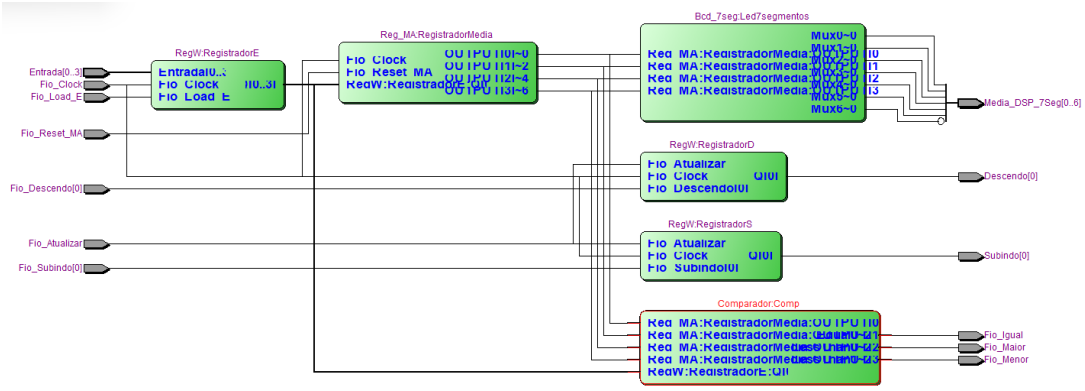

Figura 2:Diagrama Datapath

Seguiu-se para a compilação do testbench criado, este chamado de tb_datapath e que irá definir os testes do projeto. Ele está descrito abaixo:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity tb_Datapath is
end tb_Datapath;
architecture teste of tb_Datapath is
component Datapath is
port(
 Entrada : in std_logic_vector(3 downto 0);
 Fio_Load_E : in std_logic;
 Fio_Reset_MA : in std_logic;
 Fio_Descendo : in std_logic_vector(0 downto 0);
 Fio_Subindo : in std_logic_vector(0 downto 0);
 Fio_Atualizar : in std_logic;
 Fio_Clock : in std_logic;

 Fio_Maior : out std_logic;
 Fio_Igual : out std_logic;
 Fio_Menor : out std_logic;
 Subindo : out std_logic_vector(0 downto 0);
 Descendo : out std_logic_vector(0 downto 0);
 Media_DSP_7Seg : out std_logic_vector(6 downto 0)
 );
end component;
signal fio_Cl: std_logic :='0';
signal fio_Re: std_logic;
signal fio_e: std_logic_vector(3 downto 0);
signal fio_l,fio_a, fio_ma, fio_ig, fio_me: std_logic;
signal fio_di, fio_si, fio_so, fio_do: std_logic_vector(0 downto 0);
signal fio_7seg: std_logic_vector(6 downto 0);
begin
instancia_Datapath: Datapath port map(fio_e,Fio_l, fio_Re, fio_di, fio_si, fio_a, fio_cl, fio_ma, fio_ig, fio_me, fio_so, fio_do,fio_7seg);
-- Dados de entrada de 4 bits sÃ£o expressos em "hexadecimal" usando "x":
fio_Cl<= not fio_Cl after 5ns;
fio_Re<='1','0' after 20ns;
fio_l<='0','1' after 15ns;
fio_a<='0','1' after 15ns;
fio_e<=x"3",x"4" after 35ns,x"2" after 75ns;
fio_si(0)<=fio_ma;
fio_di(0)<=fio_me;

end teste;
```

Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

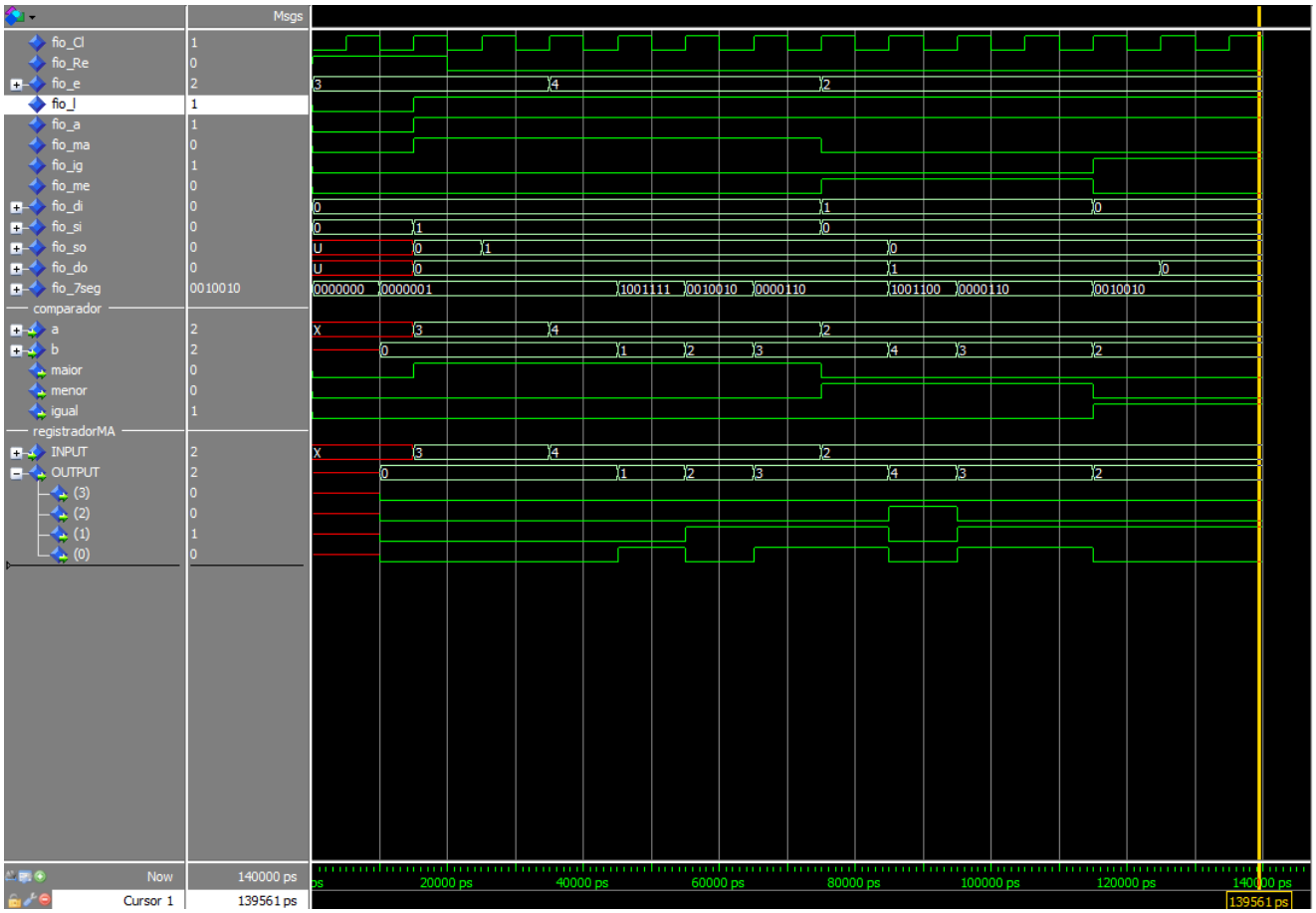

Figura 3: Simulação Datapath

Depois, com metade do projeto realizado, criou-se uma nova entidade chamada Controladora onde será feito uma maquina de estados responsável por todo o controle da sinalizadora.

Segue abaixo o código criado:

```
library ieee;
use ieee.std_logic_1164.all;
entity Controladora is
port (
 CLOCK : in std_logic; -- clock input
 RESET : in std_logic; -- reset input
 MAIOR : in std_logic; -- control input
 IGUAL : in std_logic; -- data inputs
 MENOR : in std_logic;

 LOAD_E : out std_logic; -- data output
 RESET_MA : out std_logic;
 SUBINDO : out std_logic_vector(0 downto 0);
 DESCENDO : out std_logic_vector(0 downto 0);
 ATUALIZE : out std_logic
);
end Controladora;
architecture arch of Controladora is
 type estado is (s0, s1, s2, s3, s4);
 signal est_atual, est_futuro : estado;
begin
 process(CLOCK, RESET) is
 begin
  if(RESET= '1') then
   est_atual<=s0;
  elsif(rising_edge(CLOCK)) then
   est_atual<=est_futuro;
  end if;
 end process;
 process(est_atual, MAIOR, IGUAL, MENOR) is
  begin
   case est_atual is
   when s0 =>
    LOAD_E<='1';
          RESET_MA<='1';
          SUBINDO(0)<='1';
          DESCENDO(0)<='1';
          ATUALIZE<='1';
          est_futuro<=s1;

   when s1 =>
    LOAD_E<='0';
          RESET_MA<='0';
          ATUALIZE<='0';
          SUBINDO(0)<='0';
          DESCENDO(0)<='0';
```

```
   if(MENOR='1') then
            est_futuro<=s3;
          elsif(IGUAL='1') then
            est_futuro<=s4;
          elsif(MAIOR='1') then
            est_futuro<=s2;
          else
            est_futuro<=s1;
          end if;

  when s2 =>
   LOAD_E<='1';
          SUBINDO(0)<='1';
          DESCENDO(0)<='0';
          ATUALIZE<='1';
          RESET_MA<='0';
          est_futuro<=s1;

  when s3 =>
   LOAD_E<='1';
          SUBINDO(0)<='0';
          DESCENDO(0)<='1';
          ATUALIZE<='1';
          RESET_MA<='0';
          est_futuro<=s1;

          when s4 =>
          LOAD_E<='1';
          SUBINDO(0)<='0';
          DESCENDO(0)<='0';
          ATUALIZE<='1';
          RESET_MA<='0';
            est_futuro<=s1;
         end case;
 end process;
end arch;
```

Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e State Machine Viewer. Segue abaixo as imagens:

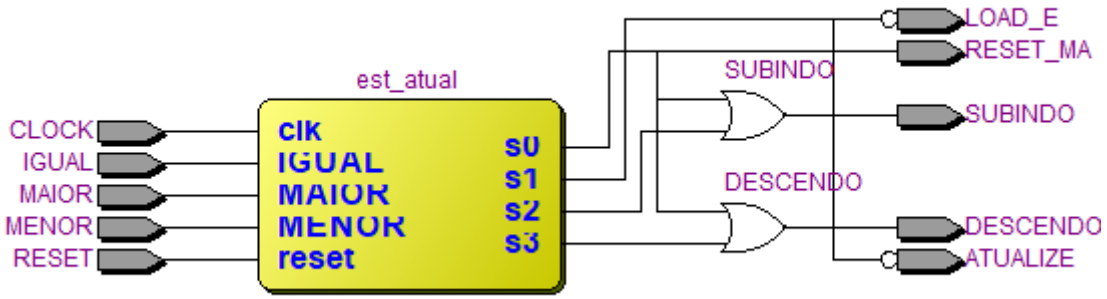

Figura 4: Circuito Controladora

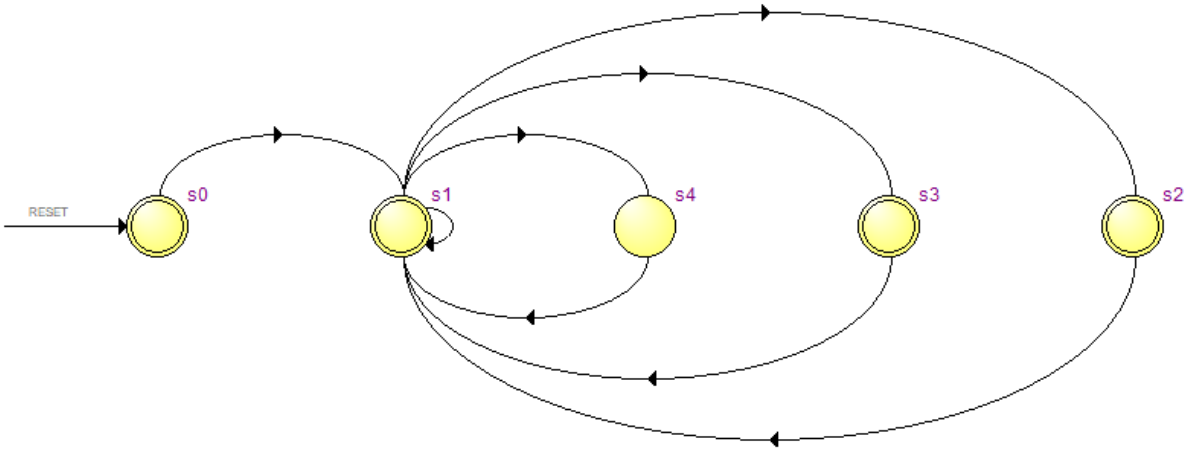

Figura 5 Máquina de Estados Controladora

Seguiu-se para a compilação do testbench criado, este chamado de tb_Controladora e que irá definir os testes do projeto. Ele está descrito abaixo:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity tb_Controladora is
end tb_Controladora;
architecture teste of tb_Controladora is
component Controladora is
port (
 CLOCK : in std_logic; -- clock input
 RESET : in std_logic; -- reset input
 MAIOR : in std_logic; -- control input
 IGUAL : in std_logic; -- data inputs
 MENOR : in std_logic;

 LOAD_E : out std_logic; -- data output
 RESET_MA : out std_logic;
 SUBINDO : out std_logic;
 DESCENDO : out std_logic;
 ATUALIZE : out std_logic
);
end component;
signal fio_clk : std_logic :='0';
signal fio_res : std_logic ;
signal fio_maior, fio_igual, fio_menor : std_logic ;
signal fio_Le,fio_ResetMa,fio_Subindo,fio_Descendo, fio_Atualize : std_logic;

begin
-- Note que o componente é instanciado com apenas 4 bits nas entradas para facilitar a simulação:
instancia_Controladora: Controladora port map(fio_clk,fio_res,fio_maior, fio_igual, fio_menor,fio_Le,fio_ResetMa,fio_Subindo,fio_Descendo, fio_Atualize);
-- Dados de entrada de 4 bits são expressos em "hexadecimal" usando "x":
fio_clk <= not fio_clk after 5ns;
fio_res <= '1', '0' after 10ns;
fio_maior <='1', '0' after 30ns;
fio_igual <='0', '1' after 50ns, '0' after 80ns;
fio_menor <='0', '1' after 110ns, '0' after 140ns;
end teste;
```

Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

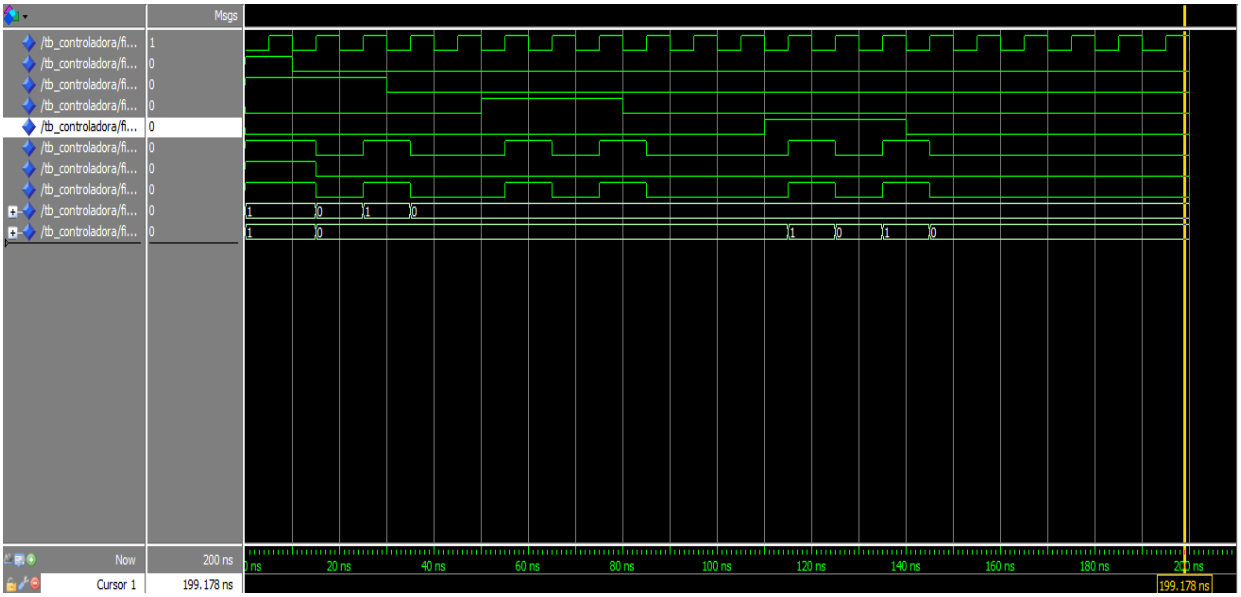

Figura 4: Simulação Controladora

Com o datapath e a controladora completos e sem erros, juntou-se os dois para a criação da FSM de alto nível sinalizador. Além disso, foi adicionado também o divisor de CLOCK de 50MHz para 1Hz para melhorar a visualização na placa, sendo que o código segue abaixo:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity Sinalizador is
 port (
 E : in std_logic_vector(3 downto 0);
 RESET : in std_logic;
 CLOCK : in std_logic;

 Subindo : out std_logic_vector(0 downto 0);
 Descendo : out std_logic_vector(0 downto 0);
 Media_DSP_7Seg : out std_logic_vector(6 downto 0)
 );
end Sinalizador;
architecture Arch of Sinalizador is
 component Controladora is
 port (
  CLOCK : in std_logic; -- clock input
  RESET : in std_logic; -- reset input
  MAIOR : in std_logic; -- control input
  IGUAL : in std_logic; -- data inputs
  MENOR : in std_logic;

  LOAD_E : out std_logic; -- data output
  RESET_MA : out std_logic;
  SUBINDO : out std_logic_vector(0 downto 0);
  DESCENDO : out std_logic_vector(0 downto 0);
  ATUALIZE : out std_logic
);
 end component;
 component Datapath is
 port (
 Entrada : in std_logic_vector(3 downto 0);
 Fio_Load_E : in std_logic;
 Fio_Reset_MA : in std_logic;
 Fio_Descendo : in std_logic_vector(0 downto 0);
 Fio_Subindo : in std_logic_vector(0 downto 0);
 Fio_Atualizar : in std_logic;
 Fio_Clock : in std_logic;

 Fio_Maior : out std_logic;
 Fio_Igual : out std_logic;
 Fio_Menor : out std_logic;
 Subindo : out std_logic_vector(0 downto 0);
 Descendo : out std_logic_vector(0 downto 0);
```

Media_DSP_7Seg : out std_logic_vector(6 downto 0)
);
end component;
component DivisorClock is
port
(
CLOCK_50MHz : in std_logic;
reset : in std_logic;
CLOCK_1Hz : out std_logic
);
end component;
signal lo_clk : std_logic;
signal Sai_Ma, Sai_Ig, Sai_Me, Ent_Lo, Ent_At, Ent_res_MA : std_logic;
signal Ent_sub, Ent_des : std_logic_vector(0 downto 0);
begin
clk: DivisorClock port map(CLOCK, RESET, lo_clk);
Controle: Controladora port map(lo_clk, RESET,Sai_Ma, Sai_Ig, Sai_Me, Ent_lo,Ent_res_MA,Ent_sub,Ent_des, Ent_at);
ViaDados: Datapath port map(E, Ent_Lo, Ent_res_MA, Ent_des, Ent_sub, Ent_at, lo_clk, Sai_Ma, Sai_Ig, Sai_Me,subindo, descendo,Media_DSP_7Seg);
end Arch;
Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e State Machine Viewer. Segue abaixo as imagens:

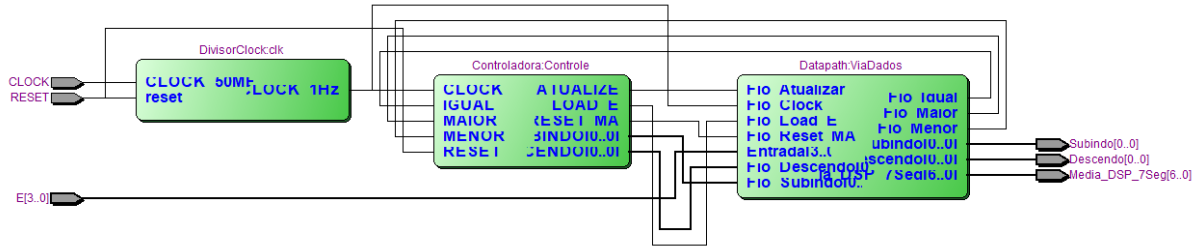

Figura 5:Circuito Sinalizador

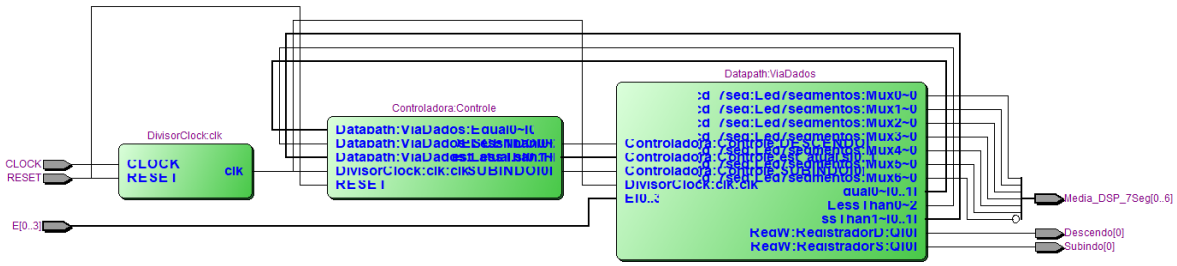

Figura 6: Diagrama Sinalizador

Seguiu-se para a compilação do testbench criado, este chamado de tb_Sinalizador e que irá definir os testes do projeto. Ele está descrito abaixo:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.numeric_std.all;
entity tb_Sinalizador is
end tb_Sinalizador;
architecture teste of tb_Sinalizador is
component Sinalizador is
port (
 E : in std_logic_vector(3 downto 0);
 RESET : in std_logic;
 CLOCK : in std_logic;

 Subindo : out std_logic_vector(0 downto 0);
 Descendo : out std_logic_vector(0 downto 0);
 Media_DSP_7Seg : out std_logic_vector(6 downto 0)
);
end component;
signal fio_clk : std_logic :='0';
signal fio_res : std_logic;
signal fio_e : std_logic_vector(3 downto 0);

signal fio_s, fio_d : std_logic_vector(0 downto 0);
signal fio_7seg : std_logic_vector(6 downto 0);

begin
-- Note que o componente é instanciado com apenas 4 bits nas entradas para facilitar a simulação:
instancia_Sinalizador: Sinalizador port map(fio_e, fio_res, fio_clk, fio_s, fio_d, fio_7seg);
-- Dados de entrada de 4 bits são expressos em "hexadecimal" usando "x":
fio_clk<=not fio_clk after 5ns;
fio_res<='1', '0' after 10ns;
fio_e<=x"0", x"8" after 50ns, x"2" after 80ns, x"1" after 100ns;

end teste;
```

Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

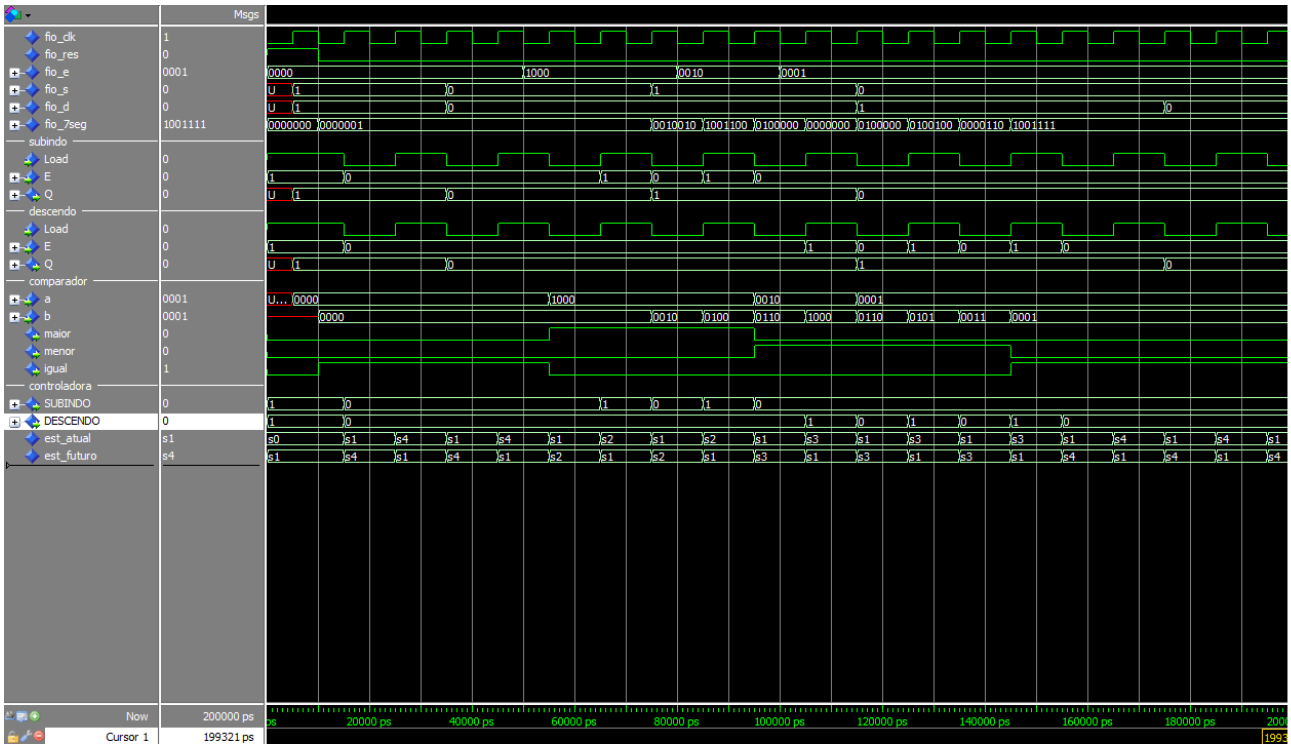

Figura 7: Simulação Sinalizador

Foi feita a pinagem do projeto:

CLOCK <=PIN_N2
Descendo[0] <=PIN_AE23
E[3] <=PIN_AE14
E[2] <=PIN_P25
E[1] <=PIN_N26
E[0] <=PIN_N25
Media_DSP_7Seg[6] <=PIN_AF10
Media_DSP_7Seg[5] <=PIN_AB12
Media_DSP_7Seg[4] <=PIN_AC12
Media_DSP_7Seg[3] <=PIN_AD11
Media_DSP_7Seg[2] <=PIN_AE11
Media_DSP_7Seg[1] <=PIN_V14
Media_DSP_7Seg[0] <=PIN_V13
RESET<=PIN_AD13
Subindo[0] <=PIN_AB21

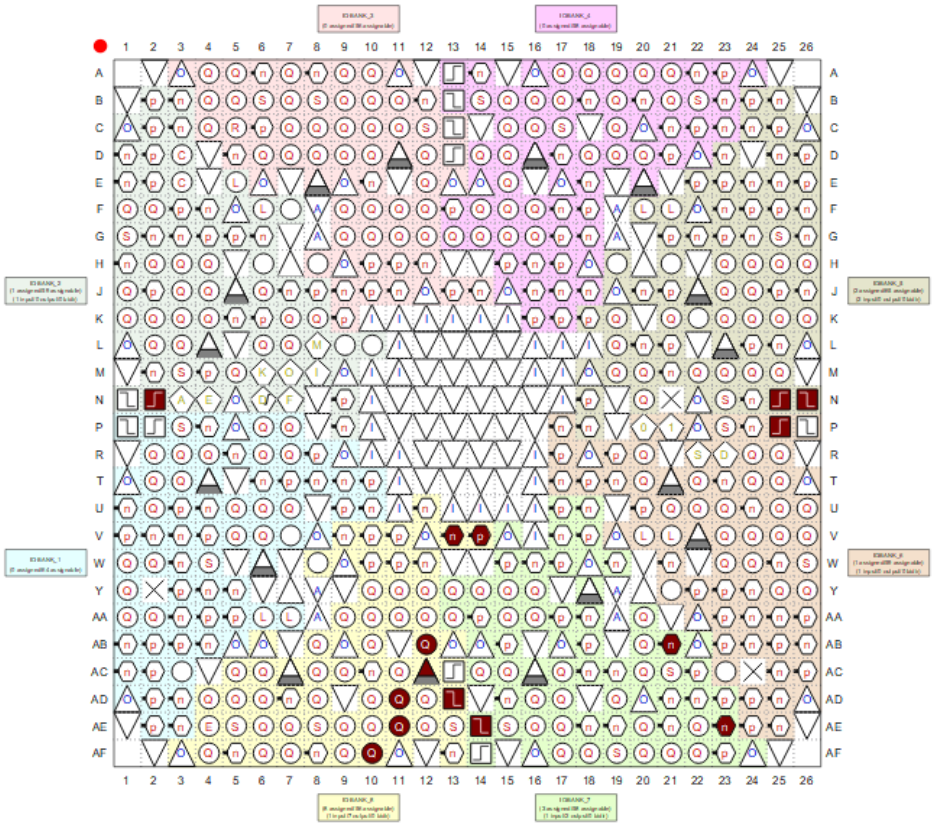

Figura 8:Pin Planner

Simulando novamente para verificação dos pinos, podemos enviar o projeto para o FPGA usando a função PROGRAMMER e o usb blaster.

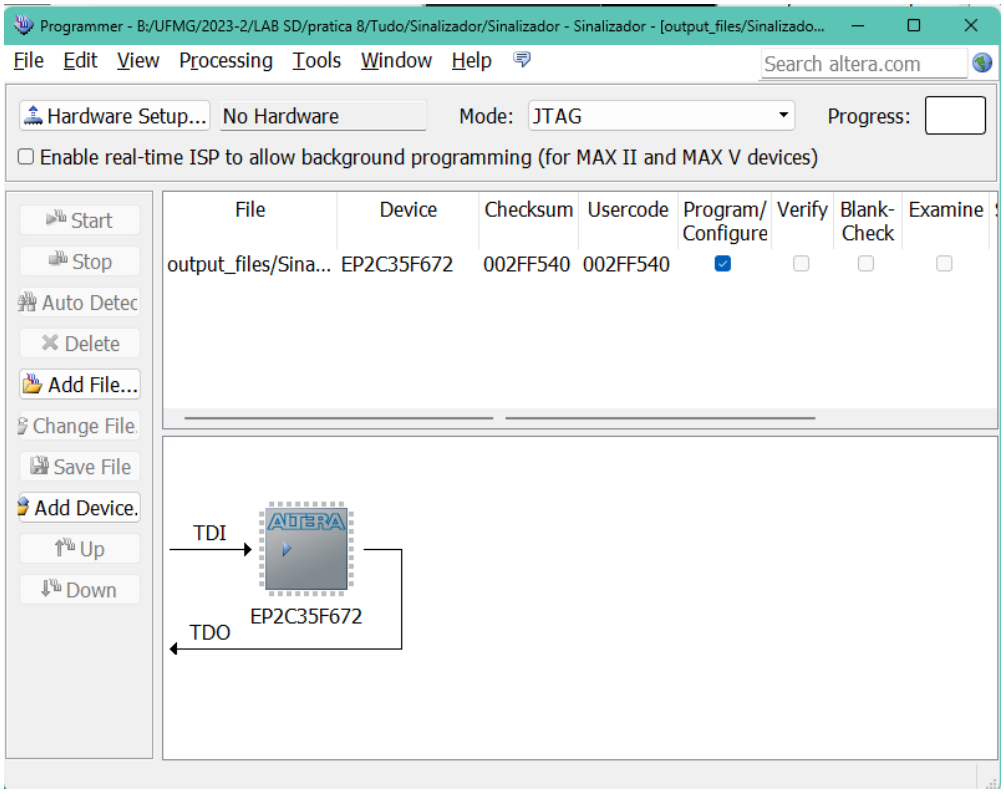

Figura 9: Programmer