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:

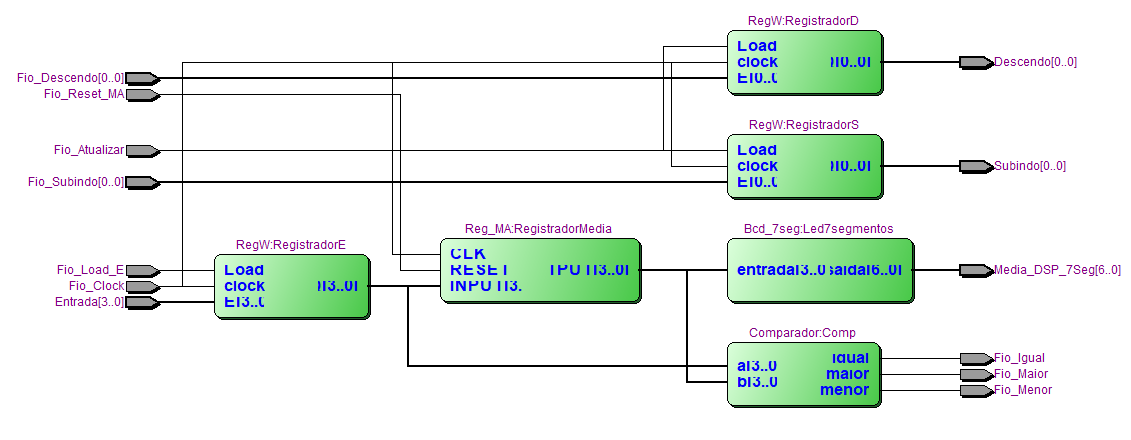


Figura :Circuito Datapath

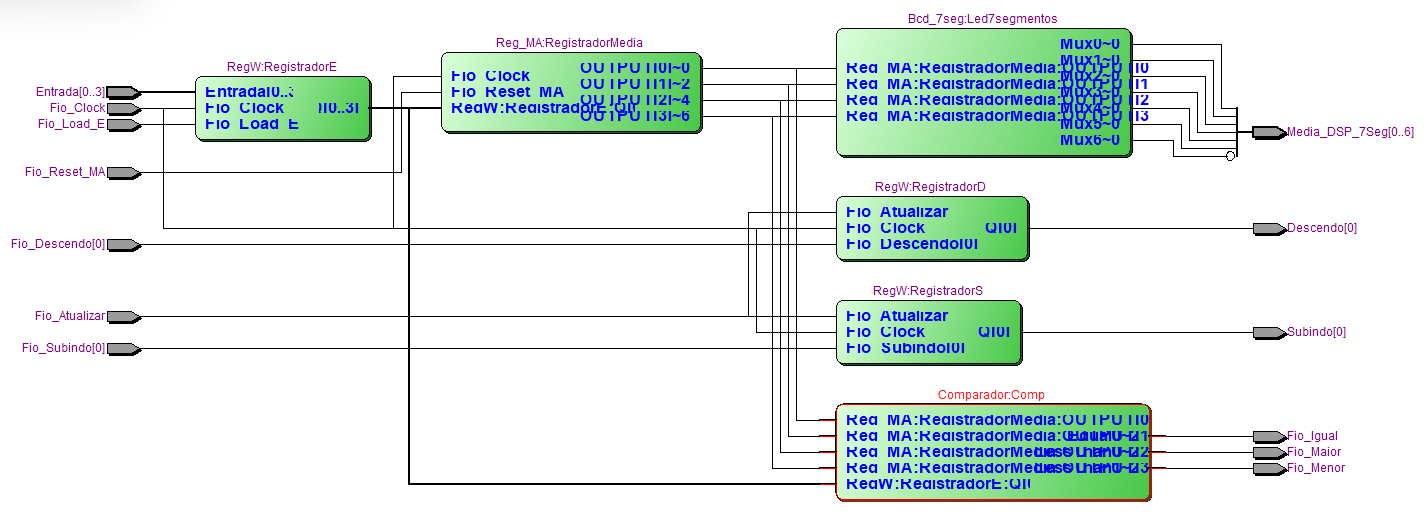


Figura :Diagrama Datapath

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.

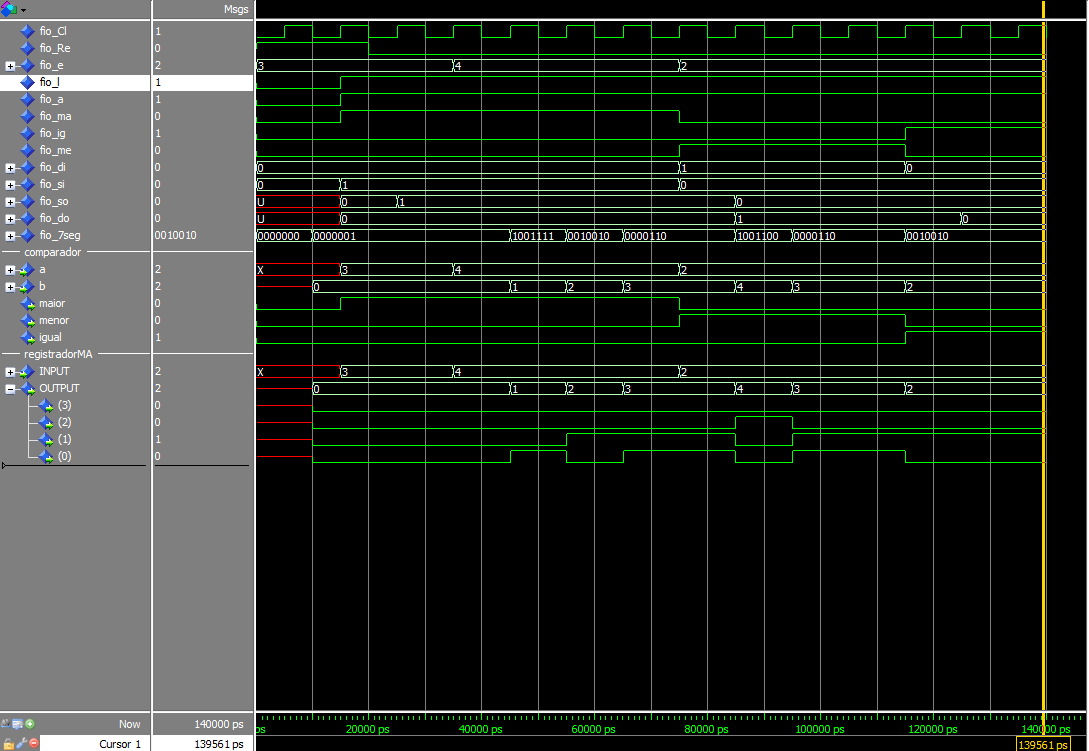


Figura : Simulação Datapath

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;***

***Diagrama

Descrição gerada automaticamenteDiagrama, Esquemático

Descrição gerada automaticamente***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 Máquina de Estados Controladora

Figura 4: Circuito 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;***

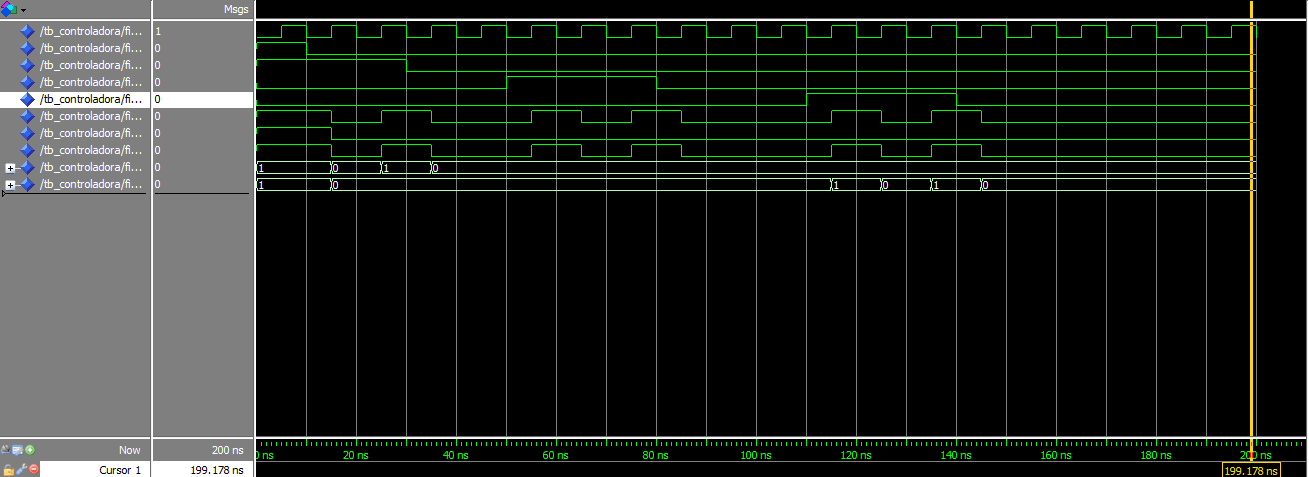
Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

Figura : 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:

Diagrama

Descrição gerada automaticamente

Figura :Circuito Sinalizador

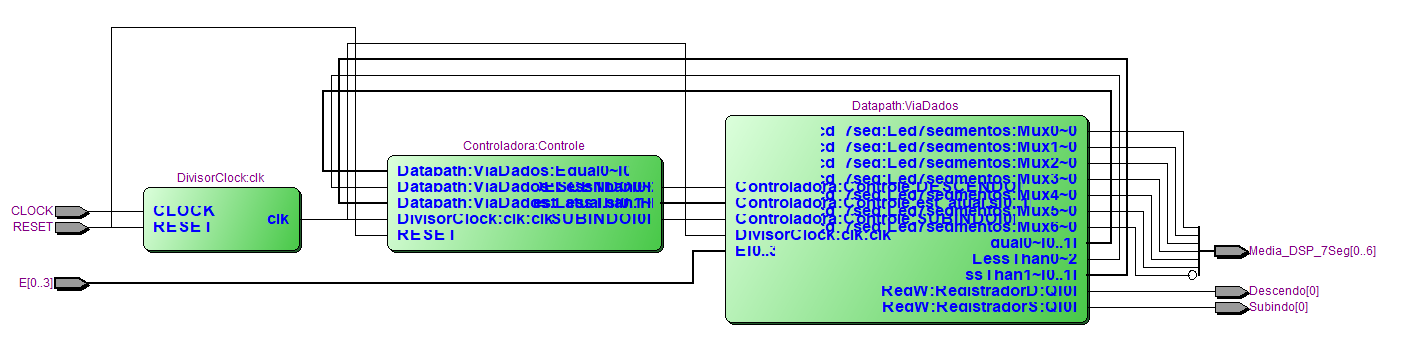


Figura : Diagrama Sinalizador

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;

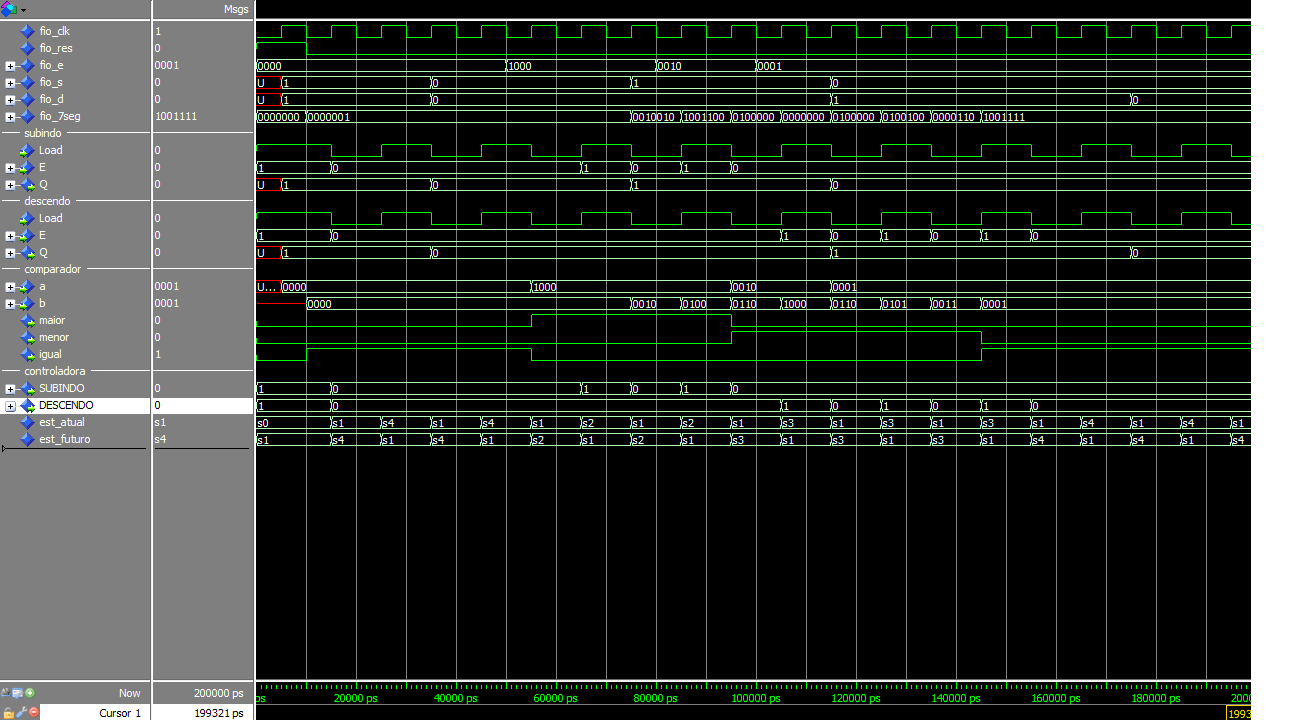
 Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

Figura : 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

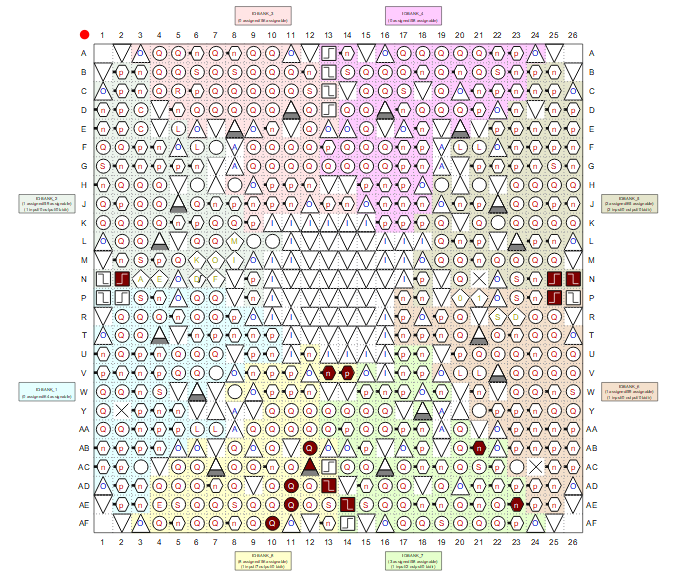


Figura :Pin Planner

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

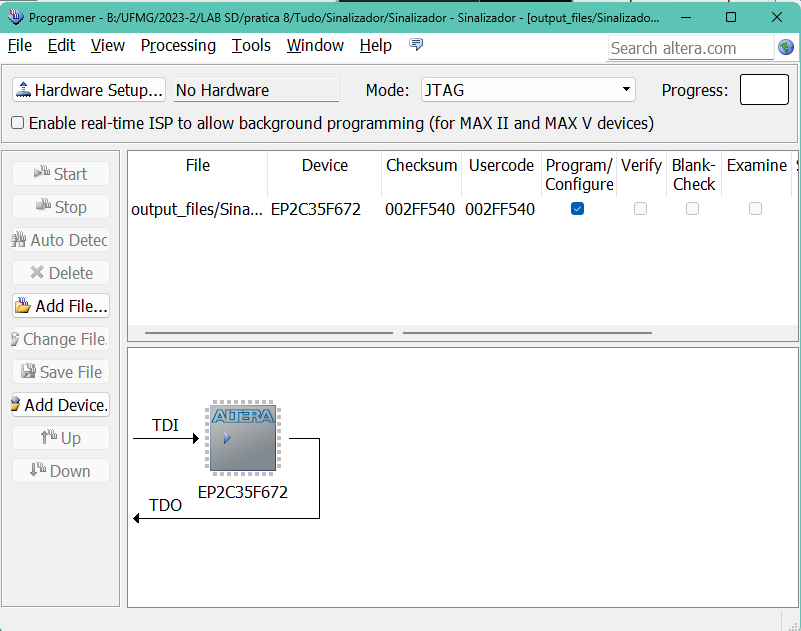


Figura : Programmer