Relatório Prática 06 – 29/09/2023
Arthur Souza/João Paulo – PN1

Iniciou a prática criando uma entidade nomeado mean_4_clocks no software QUARTUS II 13.0, especificando qual chip está sendo utilizado, no caso o EP2C35F672C6.

A partir disso, adicionou-se o código disponibilizado no sistema e que segue abaixo:

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

entity mean_4_clocks 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 mean_4_clocks;
architecture arch of mean_4_clocks is
begin
   process(CLK, RESET) is
      variable var1 : unsigned(W - 1 downto 0);
      variable var2 : unsigned(W - 1 downto 0);
      variable var3 : unsigned(W - 1 downto 0);
      variable var4 : unsigned(W - 1 downto 0);
   begin
      if (RESET = '1') then
                           var1 := to_unsigned(0,W);
                           var2 := to_unsigned(0,W);
                           var3 := to_unsigned(0,W);
                           var4 := to_unsigned(0,W);
      elsif (rising_edge(CLK)) then
         var1 := unsigned("00" & INPUT(W-1 downto 2));
         var2 := var1;
         var3 := var2;
         var4 := var3;
      end if;

      OUTPUT <= std_logic_vector(var1 + var2 + var3 + var4);
   end process;

end arch;
```

Ele apresenta 2 erros fundamentais, o primeiro sendo o uso de variables, que liga todos registradores ao input, e o segundo diz respeito a precisão do calculo e a perda de informações. Ambos foram solucionados com o código abaixo:

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

entity mean_4_clocks 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 mean_4_clocks;
architecture arch of mean_4_clocks is
signal var1, var2, var3, var4 : unsigned(W - 1 downto 0);
signal soma : unsigned(W+1 downto 0);
begin
        process(CLK, RESET) is
        begin
                if (RESET = '1') then
                        var1 <= to_unsigned(0,W);
                        var2 <= to_unsigned(0,W);
                        var3 <= to_unsigned(0,W);
                        var4 <= to_unsigned(0,W);
                elsif (rising_edge(CLK)) then
                        var1 <= unsigned(INPUT(W-1 downto 0));
                        var2 <= var1;
                        var3 <= var2;
                        var4 <= var3;
                end if;
                soma<=("00"&var1) + ("00"&var2) + ("00"&var3) + ("00"&var4);
                OUTPUT <= std_logic_vector(soma(W+1 downto 2));
        end process;
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:

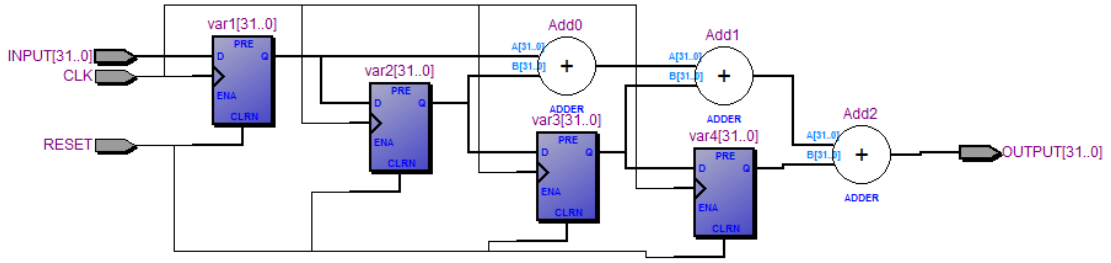

Figura 1: Circuito Mean_4_Clockr

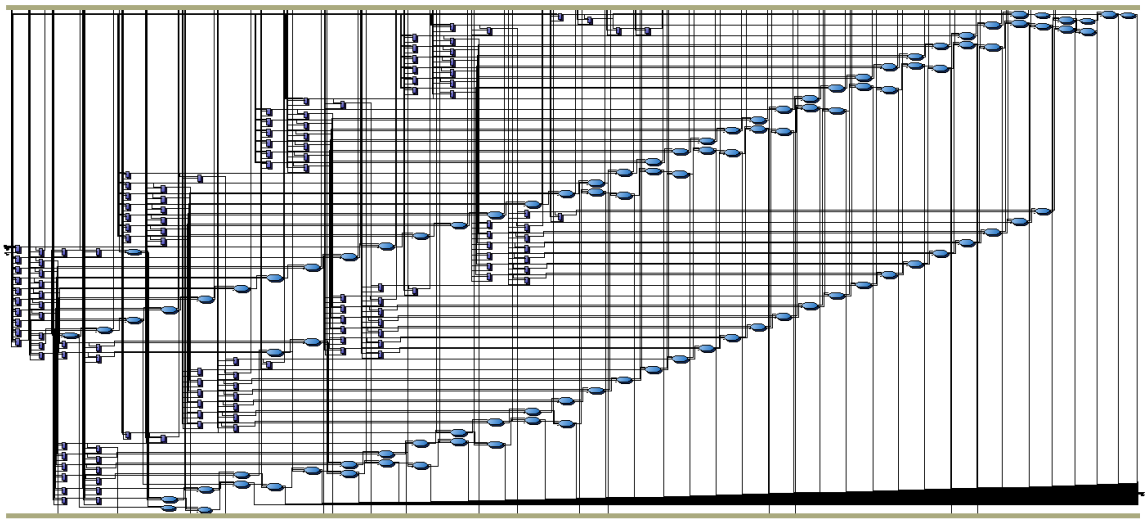

Figura 2: Diagrama do circuito Mean_4_Clock

Seguiu-se para a compilação do testbench criado, este chamado de tb_mean_4_clocks, 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_mean_4_clocks is
end tb_mean_4_clocks;

architecture teste of tb_mean_4_clocks is
component mean_4_clocks is
        generic (
      W  :  natural := 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;

signal fio_clk: std_logic:='0';
signal fio_R: std_logic;
signal fio_I, fio_O: std_logic_vector(3 downto 0);

begin
instancia_mean4clocks:  mean_4_clocks  generic  map  (W=>4)  port
map(CLK=>fio_clk, RESET=>fio_R, INPUT=>fio_I, OUTPUT=>fio_O);
-- x nas próximas linhas: os vetores de bits estão expressos em base hexadecimal
fio_clk <=not fio_clk after 25ns;
fio_R <= '1', '0' after 5ns;
fio_I <= x"1", x"f" after 300ns;
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.

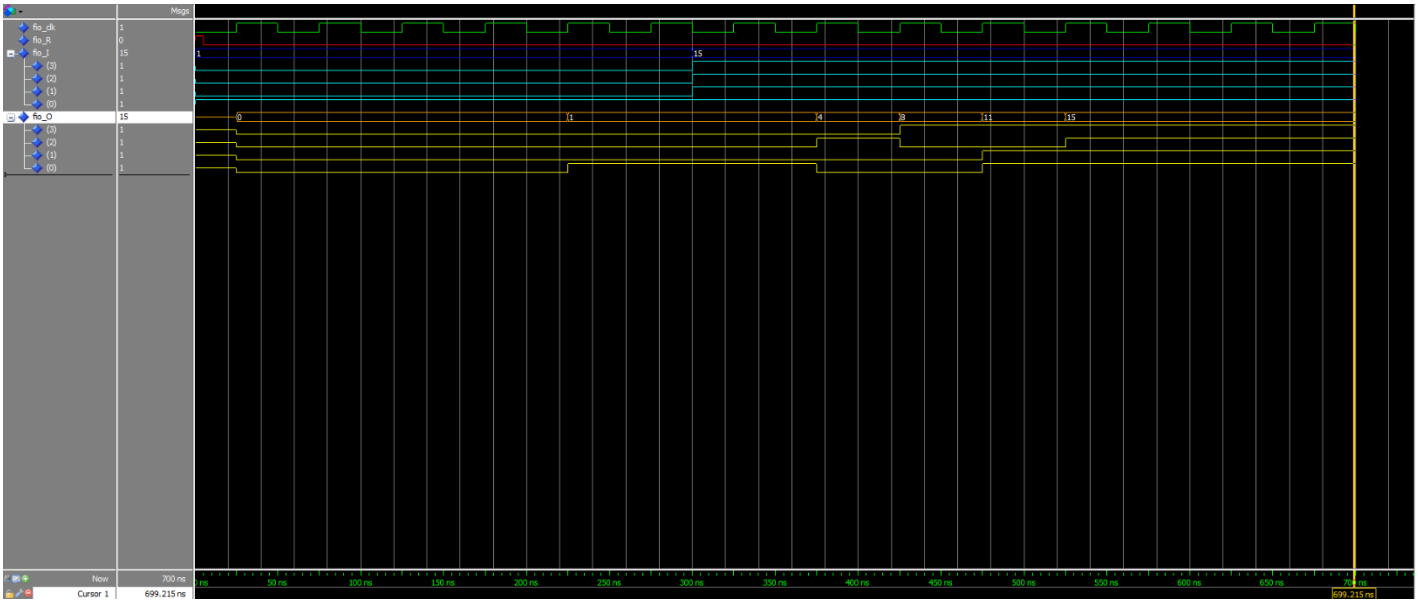

Figura 3: Simulação Mean_4_Clocks no Multisim

Foi utilizado o arquivo csv do comparador para facilitar a pinagem desse projeto.
INPUT[3] <= PIN_N25
INPUT[2] <= PIN_N26
INPUT[1] <= PIN_P25
INPUT[0] <= PIN_AE14

CLOCK <= PIN_AC13
RESET <= PIN_C13

OUTPUT[3] <= PIN_AD21
OUTPUT 2] <= PIN_AC21
OUTPUT [1] <= PIN_AA14
OUTPUT [0] <= PIN_Y13

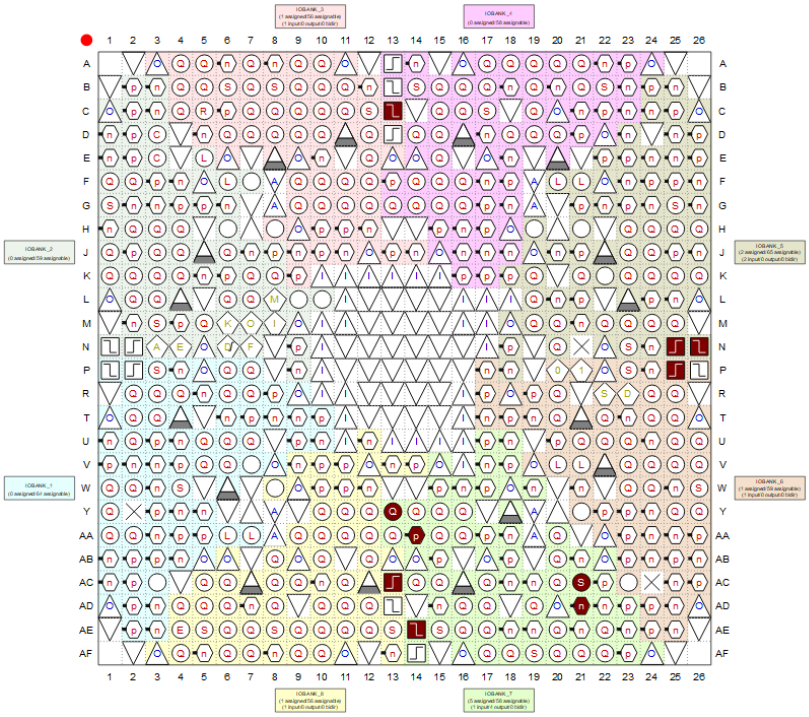

Figura 4: Pin Planner

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

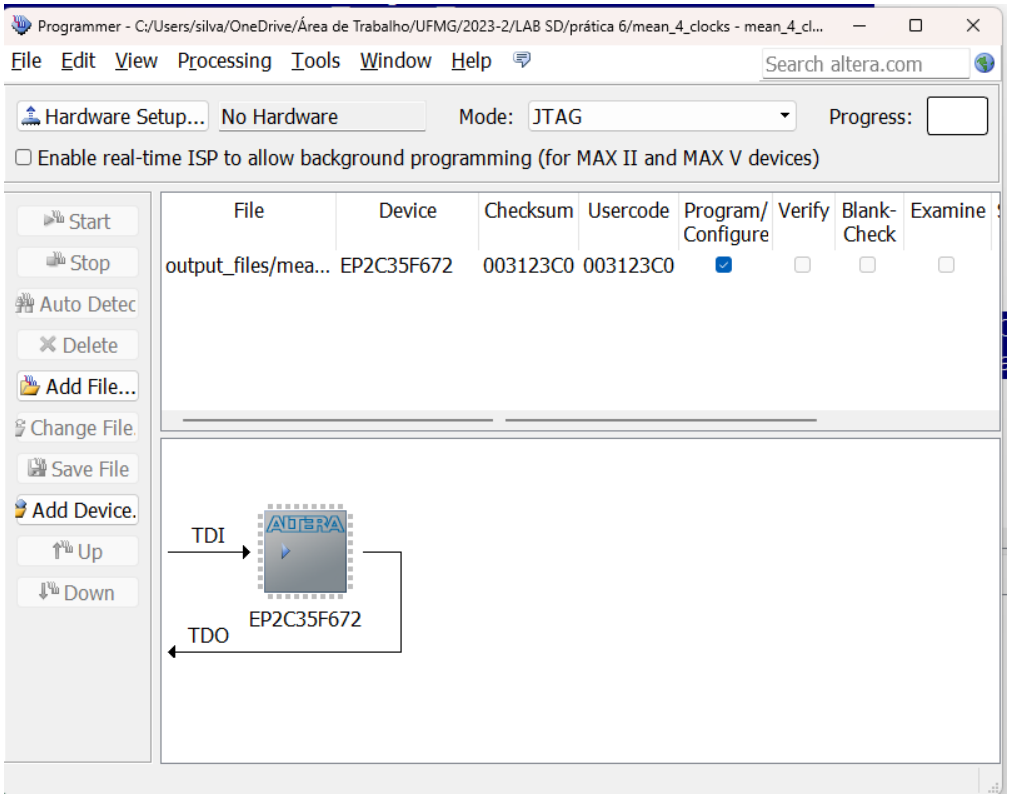

Figura 5: Programmer