Universidade de Brasília

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Organização e Arquitetura de Computadores - Turma C

Relatório
Trabalho V: Projeto do Banco de Registradores do MIPS

Aluno:
Lucas Santos - 14/0151010

Professor:
Ricardo Jacobi

28 de outubro de 2016

1 Objetivo

Projetar, simular e sintetizar um banco de registradores similar ao utilizado no MIPS, no ambiente *Quartus / ModelSim-Altera*.

2 Características do Banco de Registradores

- 32 registradores de 32 bits;
- Dois barramentos de leitura;
- Um barramento para escrita de dado, que depende de habilitação;
- Registrador 0 (índice zero) é constante, não pode ser alterado. Qualquer leitura deste registrador retorna o valor zero e escritas não afetam o seu valor.

3 Interface do Banco de Registradores

- ENTRADAS
  - **Clk**: relogio do circuito;
  - **WrEn**: habilitação de escrita. Ao ser acionado, o registrador endereçado por *WAdd* é escrito com o valor presente no barramento *WData* na transicao de subida do relógio;
  - **WData**: valor a ser escrito no registrador enderecado por *Wadd*, 32 bits;
  - **RAdd1**: endereço do registrador a ser lido em *R1*, 5 bits;
  - **RAdd2**: endereço do registrador a ser lido em *R2*, 5 bits.
- SAÍDAS
  - **R1**: porta de saída para leitura do registrador endereçado por *RAdd1*;
  - **R2**: porta de saída para leitura do registrador endereçado por *RAdd2*.

4 Descrição do trabalho

O código desenvolvido para a resolução do problema foi relativamente simples. Se ocorrer um borda de subida no *Clk*, a habilitação de escrita é verificada, se a mesma estiver ativa, o conteúdo *WData* é escrito no registrador[*WAdd*]. Independente da habilitação de escrita ou do *Clk*, o processo de leitura sempre é efetuado, sendo que o conteúdo do registrador[*RAdd1*] é copiado para a saída *R1* e o conteúdo do registrador[*RAdd2*] é copiado para a saída *R2*.

O *Test Bench* foi desenvolvido de acordo com os 3 casos de testes requisitados, sendo eles:

- 1: Escrever uma sequencia de valores 1, 2, 3, 4 ... 31 nos registradores do banco e, após, realizar a leitura de todos os registradores, em todas as combinações possíveis:
  - Leitura em *R1*;
  - Leitura em *R2*;
  - Leitura simultânea em ambos (*R1* e *R2* apresentam o mesmo registrador).
- 2: Escrever no registrador zero e verificar se ele não é alterado;
- 3: Escrita e leitura no mesmo ciclo, do mesmo registrador.

5 Códigos

5.1 Código do Banco de Registradores

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

-- Trabalho 5: Organizacao e Arquitetura de Computadores
-- Lucas Nascimento Santos Souza - 14/0151010

-- Projeto de um Banco de Registradores em VHDL
--         # 32 registradores de 32 bits;
--         # Dois barramentos de leitura;
--         # Um barramento para escrita de dado;
--         # Registrador 0 (índice zero) é constante, não pode ser alterado. Qualquer leitura deste
-- registrador retorna o valor zero e escritas não afetam o seu valor.

-- Formato do Banco de Registradores
entity BREG_OAC is
        -- WSIZE: o tamanho da palavra do banco de registradores;
        generic (WSIZE : natural := 32);
        port ( clk, wren : in std_logic;
                radd1, radd2, wadd : in std_logic_vector(4 downto 0);
                wdata : in std_logic_vector(WSIZE-1 downto 0);
                r1, r2 : out std_logic_vector(WSIZE-1 downto 0));
end BREG_OAC;

-- ENTRADAS
--         # Clk: relogio do circuito;
--         # WrEn: habilitacao de escrita. Ao ser acionado, o registrador enderecado por WAdd é escrito
-- com o valor presente no barramento WData na transicao de subida do relogio;
--        # WData: valor a ser escrito no registrador enderecado por Wadd, 32 bits;
--        # RAdd1: endereco do registrador a ser lido em R1, 5 bits;
--        # RAdd2: endereco do registrador a ser lido em R2, 5 bits;

-- SAIDAS
--        # R1: porta de saida para leitura do registrador enderecado por RAdd1;
--        # R2: porta de saída para leitura do registrador enderecado por RAdd2.

architecture comportamento of BREG_OAC is
        -- 32 registradores de 32 bits inicializados com 0
        type registradores is array(WSIZE-1 downto 0) of std_logic_vector(WSIZE-1 downto 0);
        signal registrador : registradores := ((others => (others => '0')));

begin

processo_breg: process (Clk, RAdd1, RAdd2) begin

        -- Barramentos de leitura
        -- R1 <= registrador[RAdd1]
        R1 <= registrador(to_integer(unsigned(RAdd1)));
        -- R2 <= registrador[RAdd2]
        R2 <= registrador(to_integer(unsigned(RAdd2)));
```

```
        if(rising_edge(Clk)) then
                -- Barramento de escrita
                if ((WrEn = '1') and (to_integer(unsigned(WAdd)) /= 0)) then
                        -- registrador[WAdd] <= WData
                        registrador(to_integer(unsigned(WAdd))) <= WData;
                end if;
        end if;

end process;

end architecture;
```

5.2 Código do *Test Bench*

```
-- Copyright (C) 1991-2013 Altera Corporation
-- Your use of Altera Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Altera Program License
-- Subscription Agreement, Altera MegaCore Function License
-- Agreement, or other applicable license agreement, including,
-- without limitation, that your use is for the sole purpose of
-- programming logic devices manufactured by Altera and sold by
-- Altera or its authorized distributors.  Please refer to the
-- applicable agreement for further details.

-- ***************************************************************************
-- This file contains a Vhdl test bench template that is freely editable to
-- suit user's needs .Comments are provided in each section to help the user
-- fill out necessary details.
-- ***************************************************************************
-- Generated on "10/21/2016 12:51:51"

-- Vhdl Test Bench template for design  :  BREG_OAC
--
-- Simulation tool : ModelSim-Altera (VHDL)
--

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

ENTITY BREG_OAC_vhd_tst IS
END BREG_OAC_vhd_tst;

ARCHITECTURE BREG_OAC_arch OF BREG_OAC_vhd_tst IS
-- constants
-- signals
        SIGNAL clk : STD_LOGIC := '0';
        SIGNAL r1 : STD_LOGIC_VECTOR(31 DOWNTO 0);
        SIGNAL r2 : STD_LOGIC_VECTOR(31 DOWNTO 0);
        SIGNAL radd1 : STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL radd2 : STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL wadd : STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL wdata : STD_LOGIC_VECTOR(31 DOWNTO 0);
        SIGNAL wren : STD_LOGIC;

        COMPONENT BREG_OAC
                PORT (
                        clk : IN STD_LOGIC;
                        r1 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
                        r2 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
                        radd1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        radd2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        wadd : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
                        wdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
                        wren : IN STD_LOGIC
```

```
                );
        END COMPONENT;

BEGIN
        i1 : BREG_OAC
        PORT MAP (
-- list connections between master ports and signals
                clk => clk,
                r1 => r1,
                r2 => r2,
                radd1 => radd1,
                radd2 => radd2,
                wadd => wadd,
                wdata => wdata,
                wren => wren
        );

-- CASOS DE TESTE
--         #1 Escrever uma sequencia de valores 1, 2, 3, 4 ... 31 nos registradores do
-- banco e, apos, realizar a leitura de todos os registradores, em todas as
-- combinacoes possiveis:
--                # Leitura em r1;
--                # Leitura em r2;
--                # Leitura simultanea em ambos (r1 e r2 apresentam o mesmo registrador).
--        #2 Escrever no registrador zero e verificar se ele não é alterado;
--        #3 Escrita e leitura no mesmo ciclo, do mesmo registrador. O que acontece ?

init : PROCESS
-- variable declarations
        BEGIN
        -- code that executes only once

                -----------------------------------------------------------------------------
                -- CASO #1
                -----------------------------------------------------------------------------

                -- Escrevendo nos registradores
                WrEn <= '1';

                for RegAdd in 1 to 31 loop
                        Clk <= '0'; wait for 5 ns;
                        WData <= std_logic_vector(to_unsigned(RegAdd, 32));
                        WAdd <= std_logic_vector(to_unsigned(RegAdd, 5));
                        Clk <= '1'; wait for 5 ns;
                end loop;

                -- Lendo os registradores
                WrEn <= '0';

                for RegAdd in 1 to 31 loop
                        -- Lendo em R1
                        Clk <= '0'; wait for 5 ns;
                        RAdd1 <= std_logic_vector(to_unsigned(RegAdd, 5));
                        Clk <= '1'; wait for 5 ns;
```

```
                -- Lendo em R2
                Clk <= '0'; wait for 5 ns;
                RAdd2 <= std_logic_vector(to_unsigned(RegAdd, 5));
                Clk <= '1'; wait for 5 ns;

                -- Lendo em ambos
                Clk <= '0'; wait for 5 ns;
                RAdd1 <= std_logic_vector(to_unsigned(RegAdd, 5));
                Clk <= '1'; wait for 5 ns;
            end loop;


            ------------------------------------------------------------------------------------
            -- CASO #2
            ------------------------------------------------------------------------------------

            -- Escrevendo no reg[0] (registrador que nao pode ser alterado)
            WrEn <= '1';

            -- reg[0] <= -1
            Clk <= '0'; wait for 5 ns;
            WData <= X"FFFFFFFF"; WAdd <= "00000";
            Clk <= '1'; wait for 5 ns;

            -- Lendo o reg[0] em ambos
            Clk <= '0'; wait for 5 ns;
            RAdd1 <= "00000"; RAdd2 <= "00000";
            Clk <= '1'; wait for 5 ns;

            -- reg[0] deve sempre ser 0, independente de uma tentativa de escrita no mesmo

            ------------------------------------------------------------------------------------
            -- CASO #3
            ------------------------------------------------------------------------------------

            -- Escritura e leitura no mesmo ciclo
            Clk <= '0'; wait for 5 ns;

            -- reg[1] <= -1
            WData <= X"FFFFFFFF"; WAdd <= "00001";

            -- Lendo o reg[1] em ambos
            RAdd1 <= "00001"; RAdd2 <= "00001";

            Clk <= '1'; wait for 5 ns;
            -- Se a escrita e a leitura foram feitas em um ciclo, o valor lido eh o
            -- valor anterior do reg[1]

            Clk <= '0'; wait for 5 ns;
            Clk <= '1'; wait for 5 ns;
            -- Se mais um ciclo acontecer, o valor atual de reg[1] eh lido

        WAIT;
END PROCESS init;
```

```
always : PROCESS
-- optional sensitivity list
-- (        )
-- variable declarations
        BEGIN
        -- code executes for every event on sensitivity list
        WAIT;
END PROCESS always;
END BREG_OAC_arch;
```

6 Simulação no *ModelSim*

6.1 Caso 1

6.1.1 Escrita

Figura 1: Primeira tela da parte de escrita do Caso 1 no *ModelSim*.

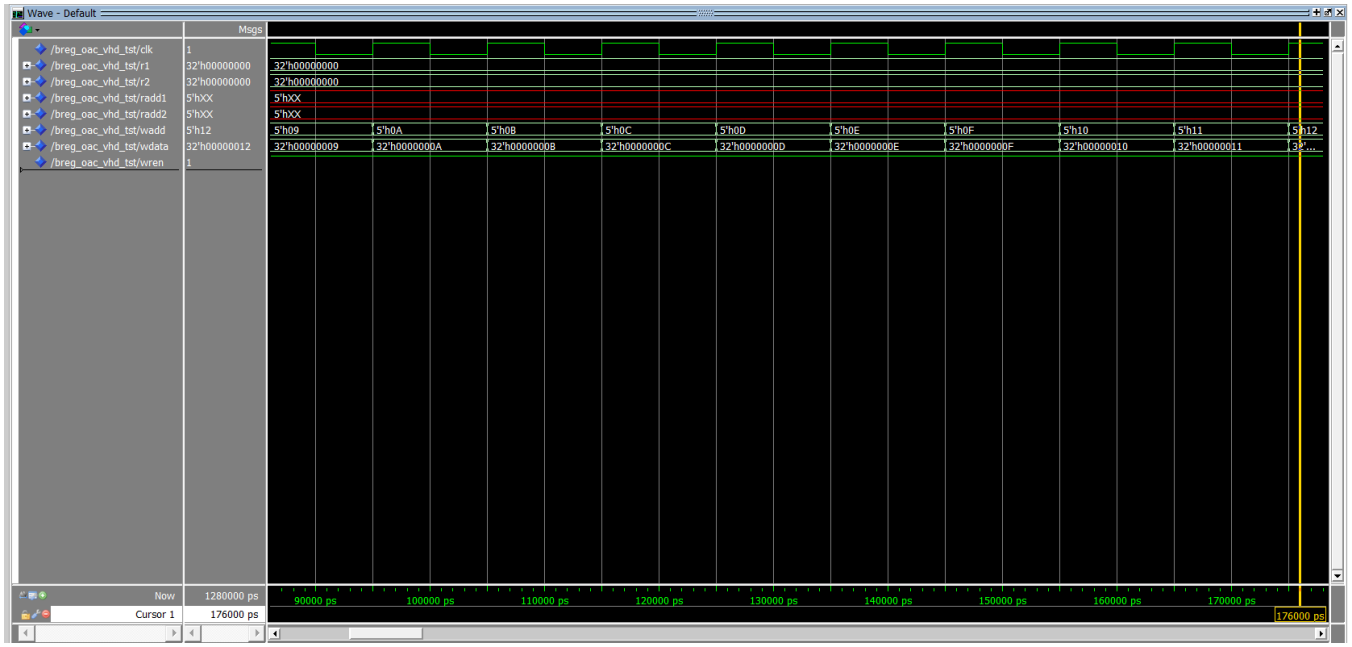

Figura 2: Segunda tela da parte de escrita do Caso 1 no *ModelSim*.

Figura 3: Terceira tela da parte de escrita do Caso 1 no *ModelSim*.

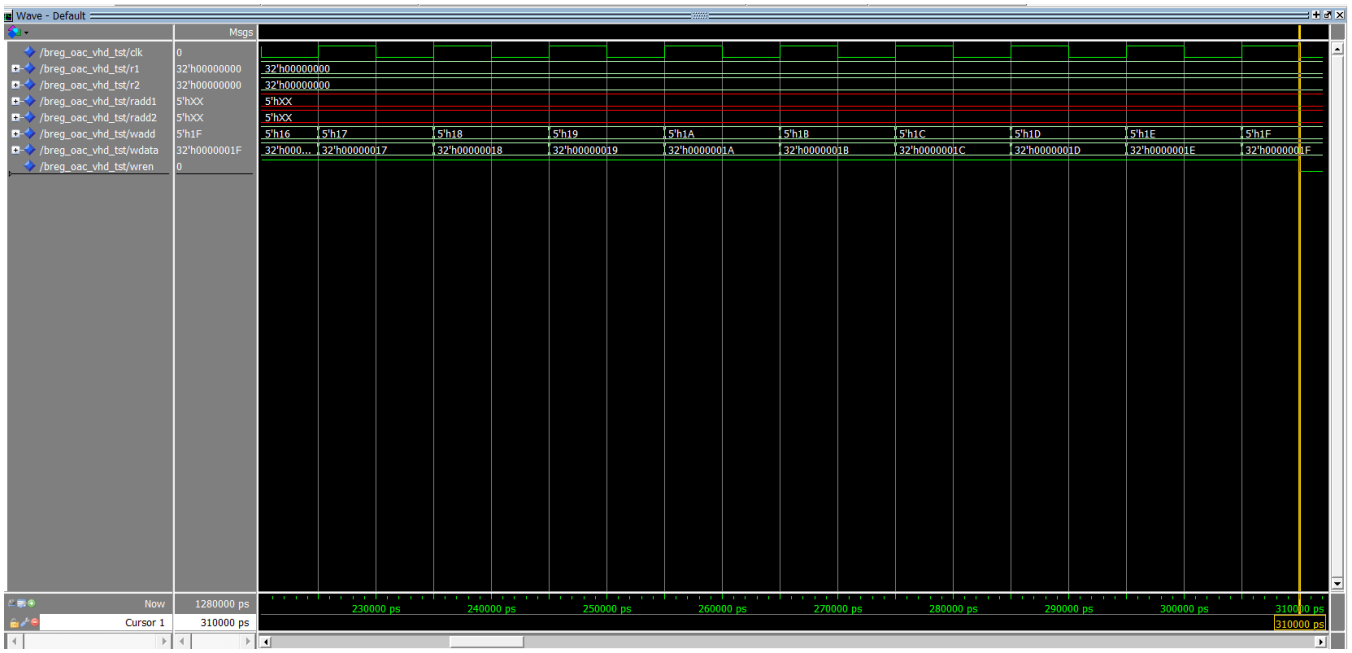

Figura 4: Última tela da parte de escrita do Caso 1 no *ModelSim*.

6.1.2 Leitura

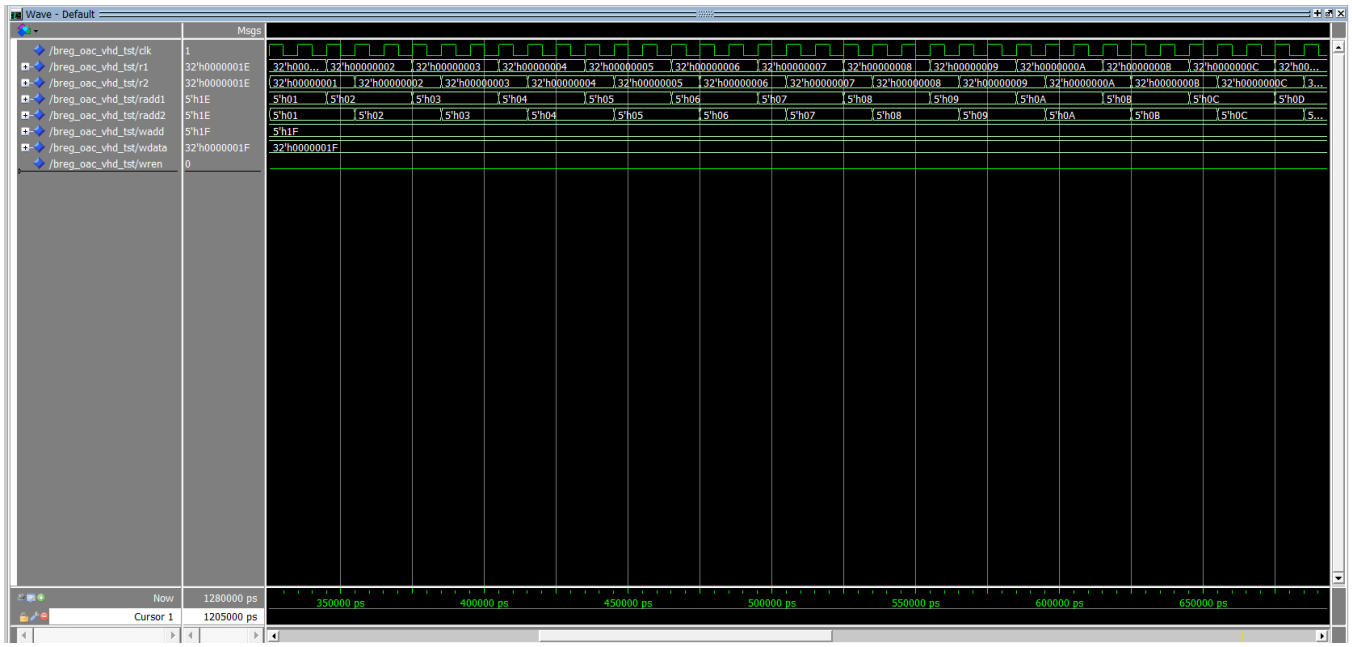

Figura 5: Primeira tela da parte de leitura do Caso 1 no *ModelSim*.

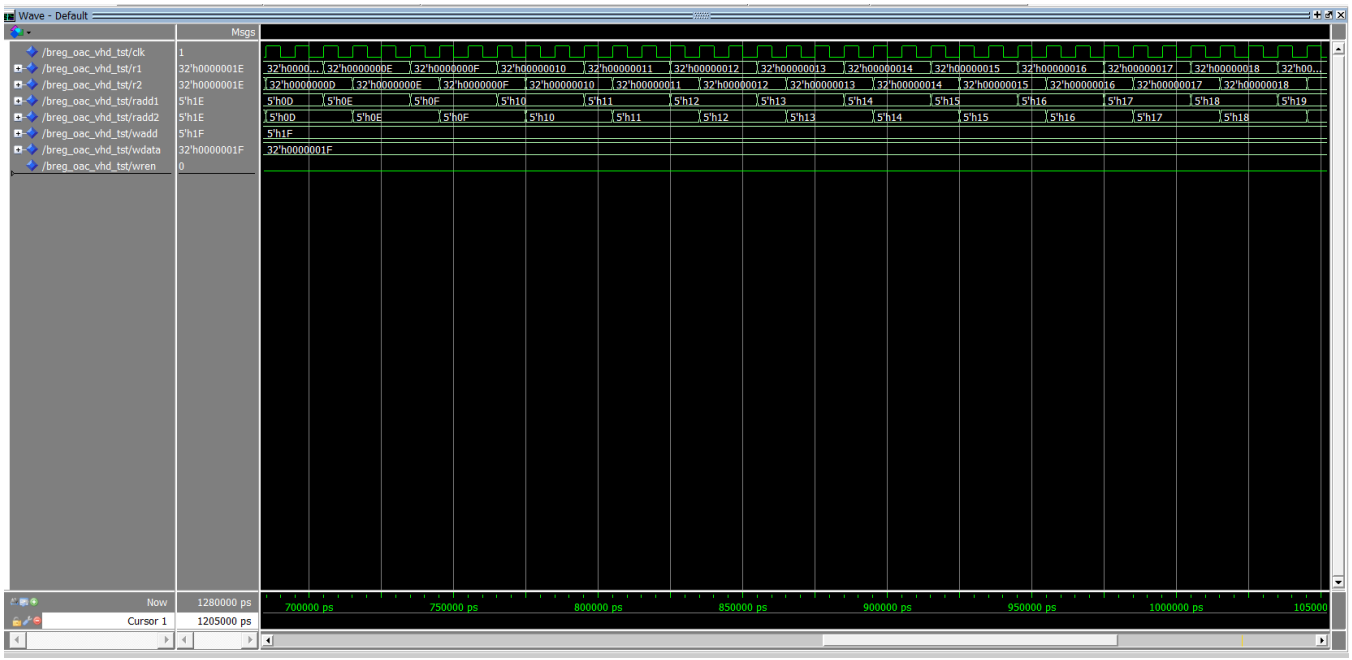

Figura 6: Segunda tela da parte de leitura do Caso 1 no *ModelSim*.

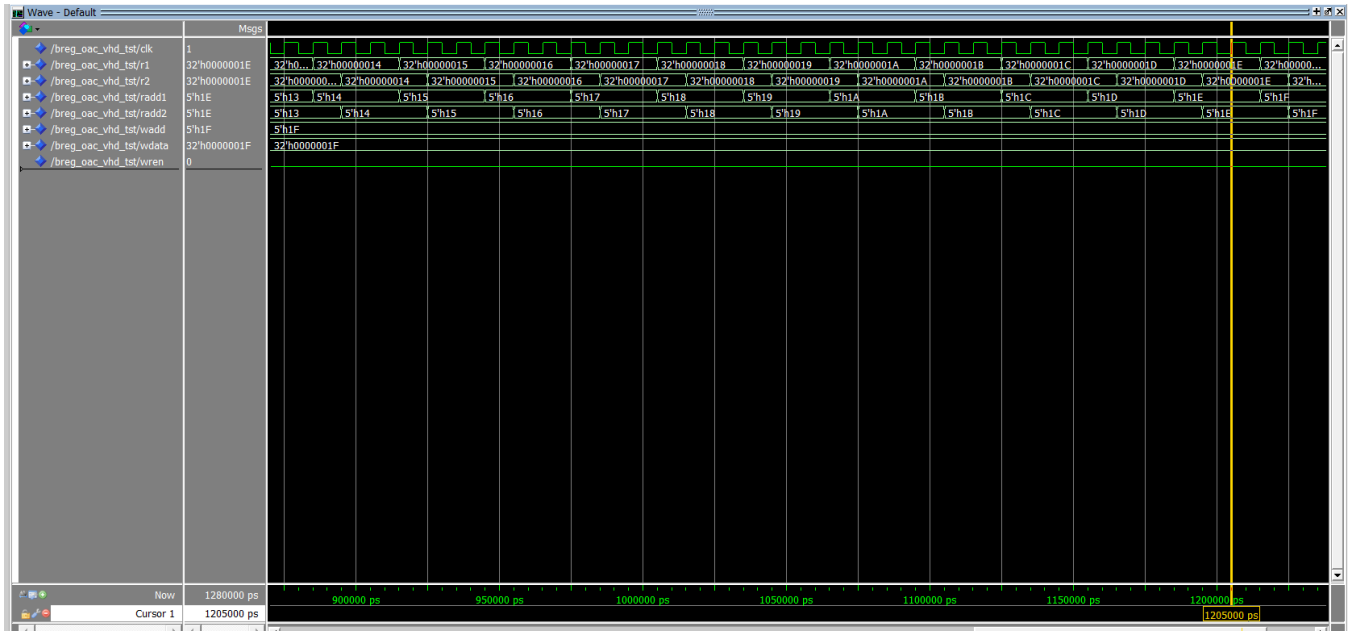

Figura 7: Última tela da parte de leitura do Caso 1 no *ModelSim*.

6.2 Caso 2 e Caso 3

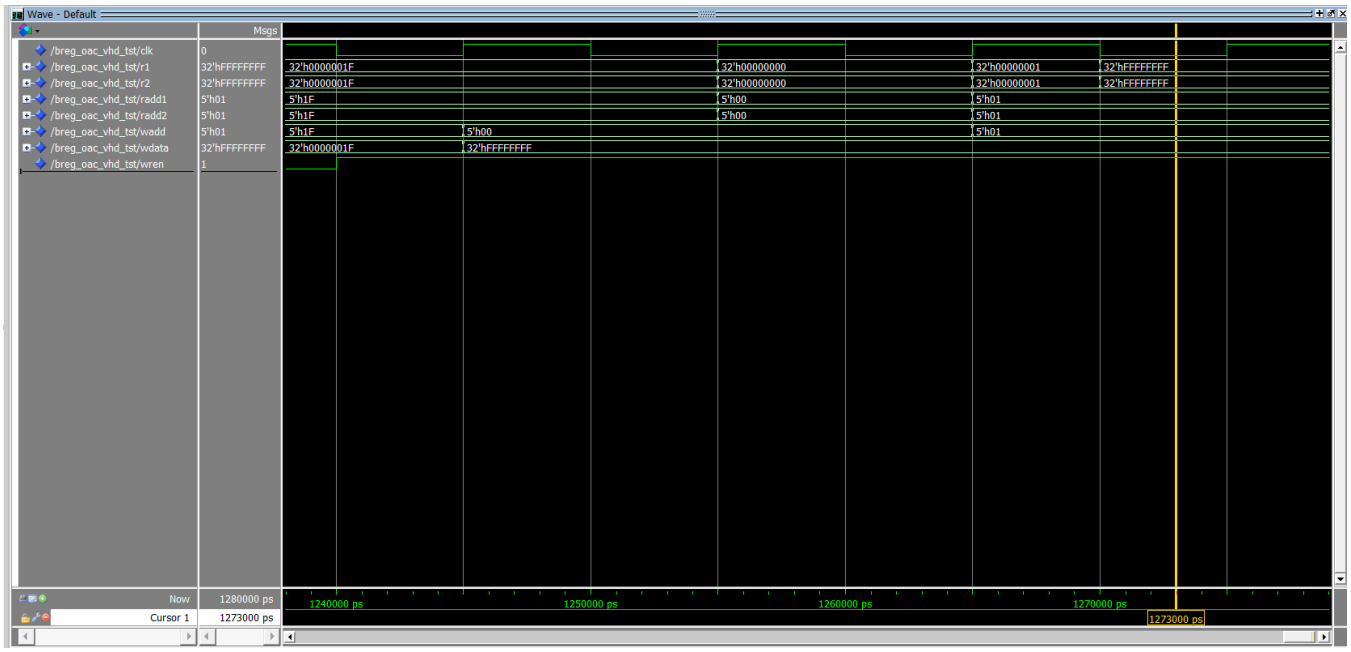

Figura 8: Tela dos Casos 2 e 3 no *ModelSim*.

7 Dados da Síntese obtidos pelo *Quartus II*

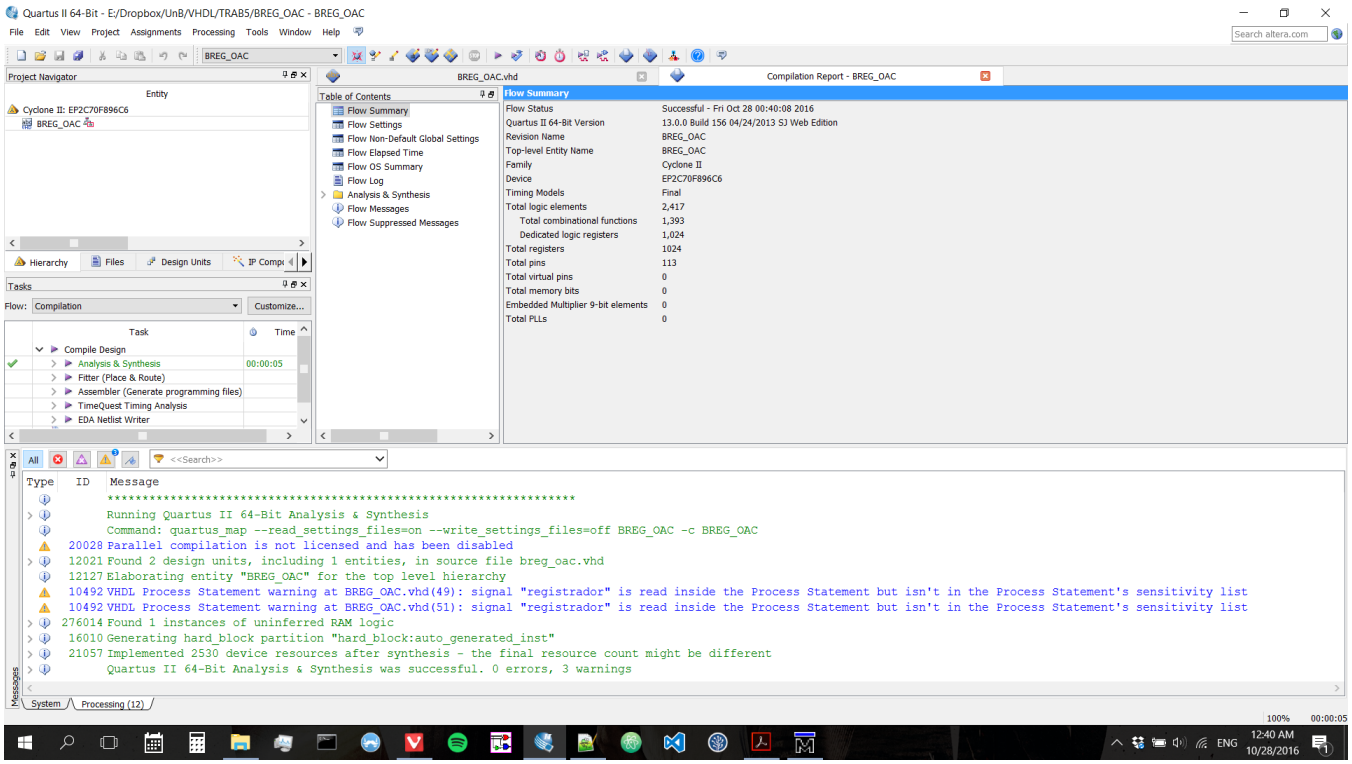

Figura 9: Dados da Sntese obtidos pelo *Quartus II* para a FPGA *Ciclone II EP270F896C6N.*