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

25 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, 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) begin

        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;
```

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

        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;

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';

                -- reg[1] <= 1
                Clk <= '0'; wait for 5 ns;
                WData <= X"00000001"; WAdd <= "00001";
                Clk <= '1'; wait for 5 ns;

                -- reg[2] <= 2
                Clk <= '0'; wait for 5 ns;
                WData <= X"00000002"; WAdd <= "00010";
                Clk <= '1'; wait for 5 ns;

                -- reg[3] <= 3
                Clk <= '0'; wait for 5 ns;
                WData <= X"00000003"; WAdd <= "00011";
                Clk <= '1'; wait for 5 ns;

                -- reg[4] <= 4
                Clk <= '0'; wait for 5 ns;
```

```
WData <= X"00000004"; WAdd <= "00100";
Clk <= '1'; wait for 5 ns;

-- reg[5] <= 5
Clk <= '0'; wait for 5 ns;
WData <= X"00000005"; WAdd <= "00101";
Clk <= '1'; wait for 5 ns;

-- reg[6] <= 6
Clk <= '0'; wait for 5 ns;
WData <= X"00000006"; WAdd <= "00110";
Clk <= '1'; wait for 5 ns;

-- reg[7] <= 7
Clk <= '0'; wait for 5 ns;
WData <= X"00000007"; WAdd <= "00111";
Clk <= '1'; wait for 5 ns;

-- reg[8] <= 8
Clk <= '0'; wait for 5 ns;
WData <= X"00000008"; WAdd <= "01000";
Clk <= '1'; wait for 5 ns;

-- reg[9] <= 9
Clk <= '0'; wait for 5 ns;
WData <= X"00000009"; WAdd <= "01001";
Clk <= '1'; wait for 5 ns;

-- reg[10] <= 10
Clk <= '0'; wait for 5 ns;
WData <= X"0000000A"; WAdd <= "01010";
Clk <= '1'; wait for 5 ns;

-- reg[11] <= 11
Clk <= '0'; wait for 5 ns;
WData <= X"0000000B"; WAdd <= "01011";
Clk <= '1'; wait for 5 ns;

-- reg[12] <= 12
Clk <= '0'; wait for 5 ns;
WData <= X"0000000C"; WAdd <= "01100";
Clk <= '1'; wait for 5 ns;

-- reg[13] <= 13
Clk <= '0'; wait for 5 ns;
WData <= X"0000000D"; WAdd <= "01101";
Clk <= '1'; wait for 5 ns;

-- reg[14] <= 14
Clk <= '0'; wait for 5 ns;
WData <= X"0000000E"; WAdd <= "01110";
Clk <= '1'; wait for 5 ns;

-- reg[15] <= 15
Clk <= '0'; wait for 5 ns;
WData <= X"0000000F"; WAdd <= "01111";
```

```
Clk <= '1'; wait for 5 ns;

-- reg[16] <= 16
Clk <= '0'; wait for 5 ns;
WData <= X"00000010"; WAdd <= "10000";
Clk <= '1'; wait for 5 ns;

-- reg[17] <= 17
Clk <= '0'; wait for 5 ns;
WData <= X"00000011"; WAdd <= "10001";
Clk <= '1'; wait for 5 ns;

-- reg[18] <= 18
Clk <= '0'; wait for 5 ns;
WData <= X"00000012"; WAdd <= "10010";
Clk <= '1'; wait for 5 ns;

-- reg[19] <= 19
Clk <= '0'; wait for 5 ns;
WData <= X"00000013"; WAdd <= "10011";
Clk <= '1'; wait for 5 ns;

-- reg[20] <= 20
Clk <= '0'; wait for 5 ns;
WData <= X"00000014"; WAdd <= "10100";
Clk <= '1'; wait for 5 ns;

-- reg[21] <= 21
Clk <= '0'; wait for 5 ns;
WData <= X"00000015"; WAdd <= "10101";
Clk <= '1'; wait for 5 ns;

-- reg[22] <= 22
Clk <= '0'; wait for 5 ns;
WData <= X"00000016"; WAdd <= "10110";
Clk <= '1'; wait for 5 ns;

-- reg[23] <= 23
Clk <= '0'; wait for 5 ns;
WData <= X"00000017"; WAdd <= "10111";
Clk <= '1'; wait for 5 ns;

-- reg[24] <= 24
Clk <= '0'; wait for 5 ns;
WData <= X"00000018"; WAdd <= "11000";
Clk <= '1'; wait for 5 ns;

-- reg[25] <= 25
Clk <= '0'; wait for 5 ns;
WData <= X"00000019"; WAdd <= "11001";
Clk <= '1'; wait for 5 ns;

-- reg[26] <= 26
Clk <= '0'; wait for 5 ns;
WData <= X"0000001A"; WAdd <= "11010";
Clk <= '1'; wait for 5 ns;
```

```
-- reg[27] <= 27
Clk <= '0'; wait for 5 ns;
WData <= X"0000001B"; WAdd <= "11011";
Clk <= '1'; wait for 5 ns;

-- reg[28] <= 28
Clk <= '0'; wait for 5 ns;
WData <= X"0000001C"; WAdd <= "11100";
Clk <= '1'; wait for 5 ns;

-- reg[29] <= 29
Clk <= '0'; wait for 5 ns;
WData <= X"0000001D"; WAdd <= "11101";
Clk <= '1'; wait for 5 ns;

-- reg[30] <= 30
Clk <= '0'; wait for 5 ns;
WData <= X"0000001E"; WAdd <= "11110";
Clk <= '1'; wait for 5 ns;

-- reg[31] <= 31
Clk <= '0'; wait for 5 ns;
WData <= X"0000001F"; WAdd <= "11111";
Clk <= '1'; wait for 5 ns;

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

-- reg[1]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00001";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00001";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00001"; RAdd2 <= "00001";
Clk <= '1'; wait for 5 ns;

-- reg[2]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00010";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00010";
Clk <= '1'; wait for 5 ns;
```

```
-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00010"; RAdd2 <= "00010";
Clk <= '1'; wait for 5 ns;

-- reg[3]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00011";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00011";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00011"; RAdd2 <= "00011";
Clk <= '1'; wait for 5 ns;

-- reg[4]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00100";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00100";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00100"; RAdd2 <= "00100";
Clk <= '1'; wait for 5 ns;

-- reg[5]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00101";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00101";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00101"; RAdd2 <= "00101";
Clk <= '1'; wait for 5 ns;

-- reg[6]
-- Lendo em R1
```

```
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00110";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00110";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00110"; RAdd2 <= "00110";
Clk <= '1'; wait for 5 ns;

-- reg[7]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00111";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "00111";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "00111"; RAdd2 <= "00111";
Clk <= '1'; wait for 5 ns;

-- reg[8]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01000";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01000";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01000"; RAdd2 <= "01000";
Clk <= '1'; wait for 5 ns;

-- reg[9]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01001";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01001";
Clk <= '1'; wait for 5 ns;
```

```
-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01001"; RAdd2 <= "01001";
Clk <= '1'; wait for 5 ns;

-- reg[10]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01010";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01010";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01010"; RAdd2 <= "01010";
Clk <= '1'; wait for 5 ns;

-- reg[11]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01011";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01011";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01011"; RAdd2 <= "01011";
Clk <= '1'; wait for 5 ns;

-- reg[12]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01100";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01100";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01100"; RAdd2 <= "01100";
Clk <= '1'; wait for 5 ns;

-- reg[13]
-- Lendo em R1
```

```
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01101";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01101";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01101"; RAdd2 <= "01101";
Clk <= '1'; wait for 5 ns;

-- reg[14]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01110";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01110";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01110"; RAdd2 <= "01110";
Clk <= '1'; wait for 5 ns;

-- reg[15]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01111";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "01111";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "01111"; RAdd2 <= "01111";
Clk <= '1'; wait for 5 ns;

-- reg[16]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10000";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10000";
Clk <= '1'; wait for 5 ns;
```

```
-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10000"; RAdd2 <= "10000";
Clk <= '1'; wait for 5 ns;

-- reg[17]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10001";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10001";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10001"; RAdd2 <= "10001";
Clk <= '1'; wait for 5 ns;

-- reg[18]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10010";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10010";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10010"; RAdd2 <= "10010";
Clk <= '1'; wait for 5 ns;

-- reg[19]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10011";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10011";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10011"; RAdd2 <= "10011";
Clk <= '1'; wait for 5 ns;

-- reg[20]
-- Lendo em R1
```

```
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10100";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10100";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10100"; RAdd2 <= "10100";
Clk <= '1'; wait for 5 ns;

-- reg[21]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10101";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10101";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10101"; RAdd2 <= "10101";
Clk <= '1'; wait for 5 ns;

-- reg[22]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10110";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10110";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10110"; RAdd2 <= "10110";
Clk <= '1'; wait for 5 ns;

-- reg[23]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10111";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "10111";
Clk <= '1'; wait for 5 ns;
```

```
-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "10111"; RAdd2 <= "10111";
Clk <= '1'; wait for 5 ns;

-- reg[24]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11000";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11000";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11000"; RAdd2 <= "11000";
Clk <= '1'; wait for 5 ns;

-- reg[25]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11001";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11001";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11001"; RAdd2 <= "11001";
Clk <= '1'; wait for 5 ns;

-- reg[26]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11010";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11010";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11010"; RAdd2 <= "11010";
Clk <= '1'; wait for 5 ns;

-- reg[27]
-- Lendo em R1
```

```
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11011";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11011";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11011"; RAdd2 <= "11011";
Clk <= '1'; wait for 5 ns;

-- reg[28]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11100";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11100";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11100"; RAdd2 <= "11100";
Clk <= '1'; wait for 5 ns;

-- reg[29]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11101";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11101";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11101"; RAdd2 <= "11101";
Clk <= '1'; wait for 5 ns;

-- reg[30]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11110";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11110";
Clk <= '1'; wait for 5 ns;
```

```
-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11110"; RAdd2 <= "11110";
Clk <= '1'; wait for 5 ns;

-- reg[31]
-- Lendo em R1
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11111";
Clk <= '1'; wait for 5 ns;

-- Lendo em R2
Clk <= '0'; wait for 5 ns;
RAdd2 <= "11111";
Clk <= '1'; wait for 5 ns;

-- Lendo em ambos
Clk <= '0'; wait for 5 ns;
RAdd1 <= "11111"; RAdd2 <= "11111";
Clk <= '1'; wait for 5 ns;


-----------------------------------------------------------------------------------------
-- 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*.

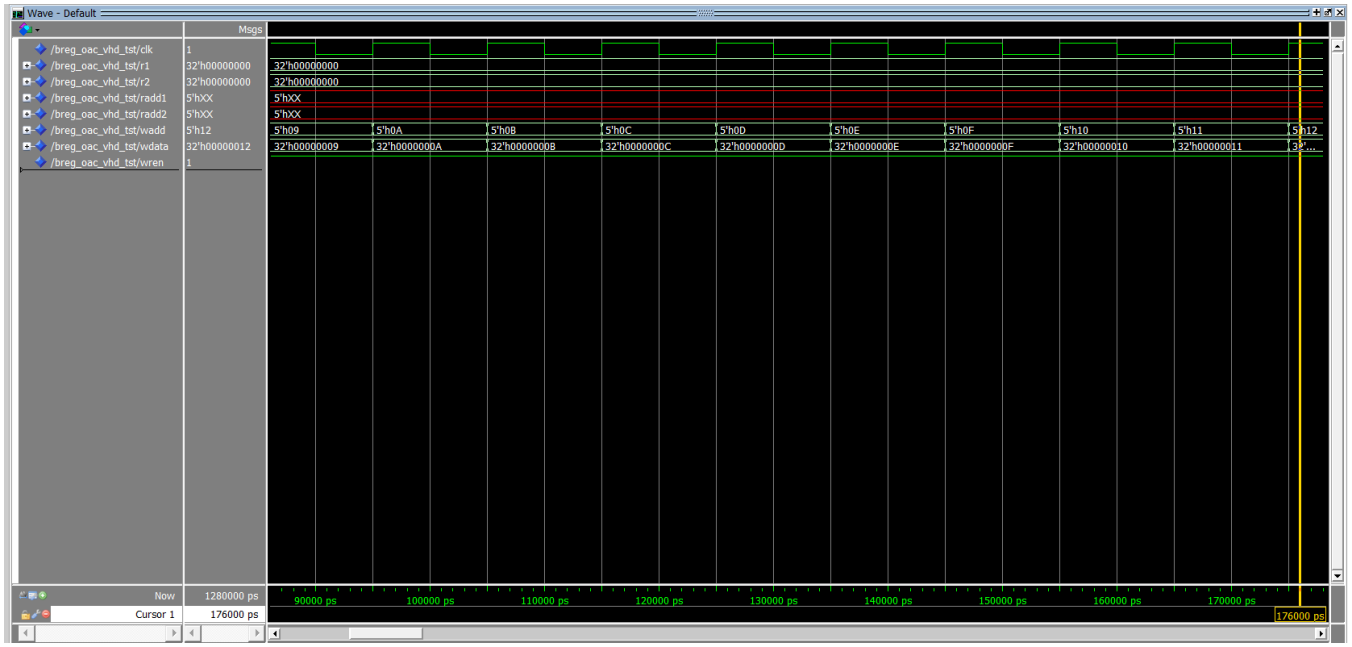

Figura 2: Segunda tela da parte de escrita do Caso 1 no *ModelSim*.

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

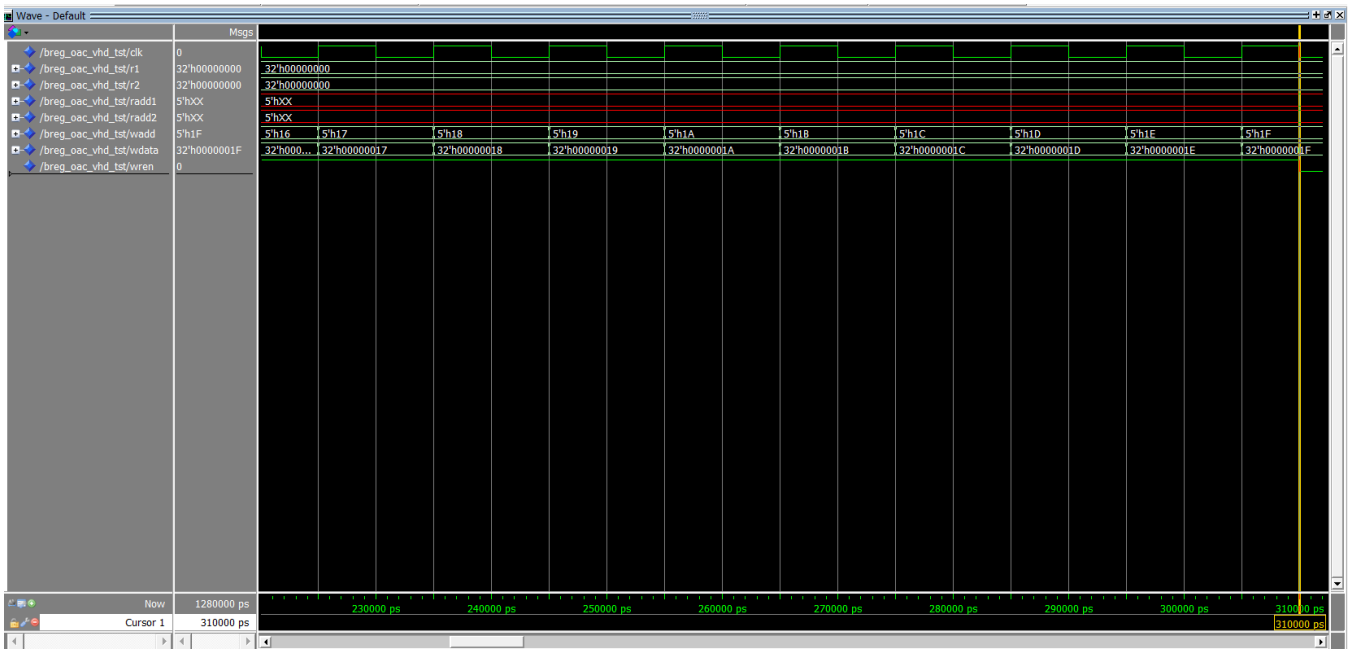

Figura 4: Última tela da parte de escrita do Caso 1 no *ModelSim*.

6.1.2 Leitura

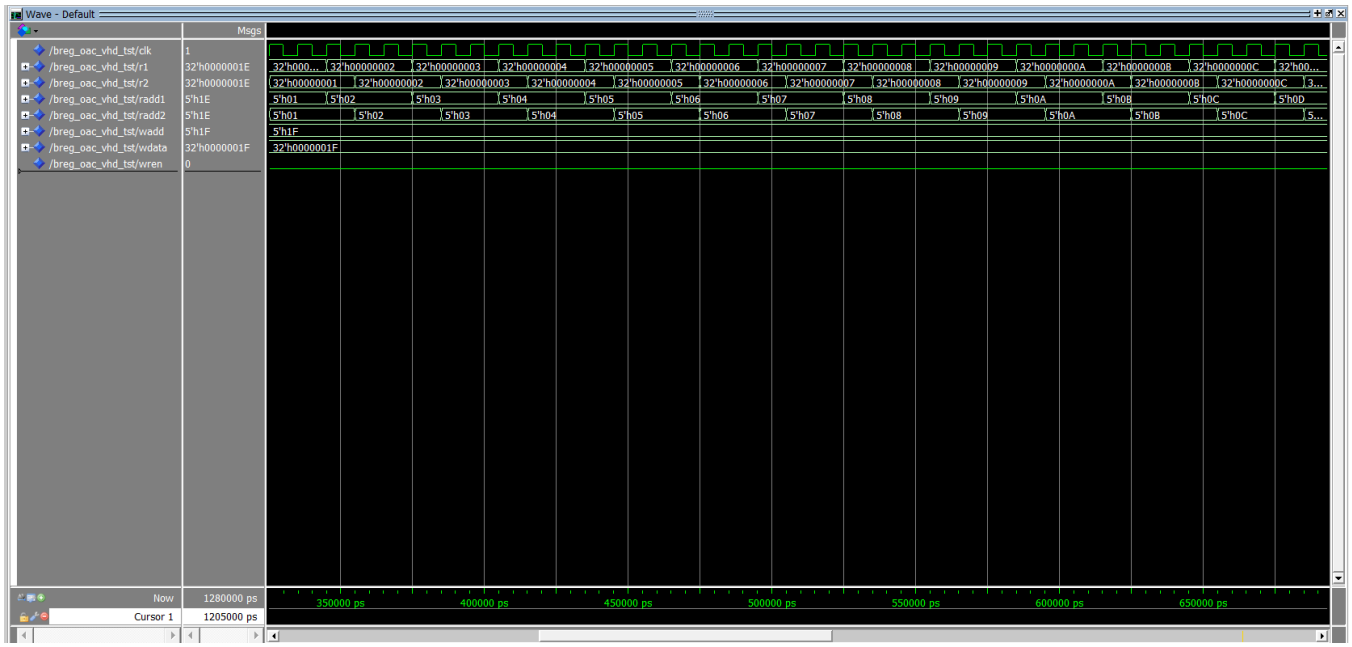

Figura 5: Primeira tela da parte de leitura do Caso 1 no *ModelSim*.

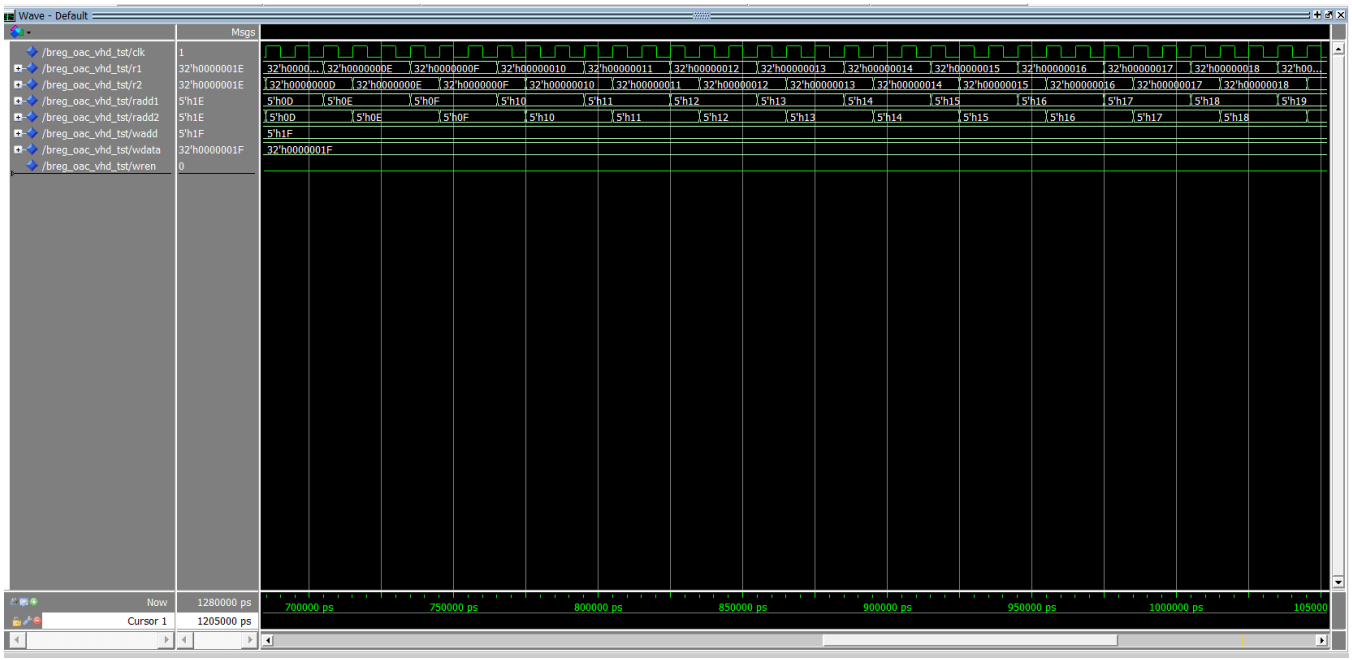

Figura 6: Segunda tela da parte de leitura do Caso 1 no *ModelSim*.

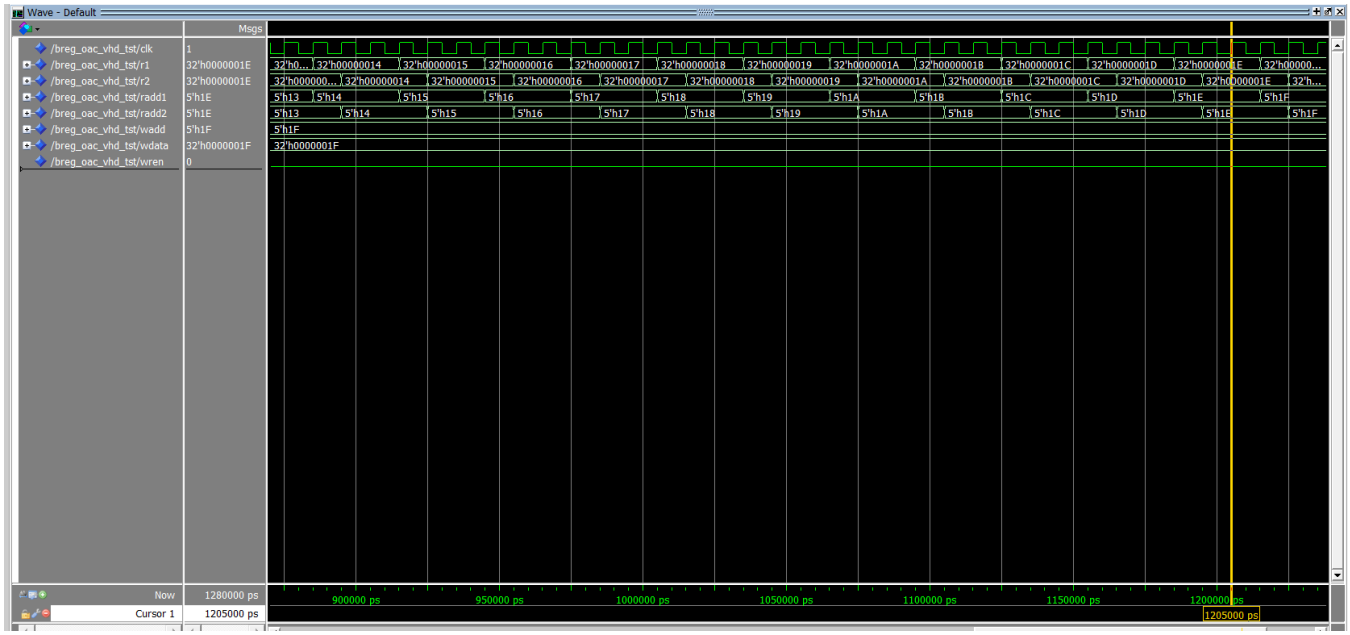

Figura 7: Última tela da parte de leitura do Caso 1 no *ModelSim*.

6.2 Caso 2

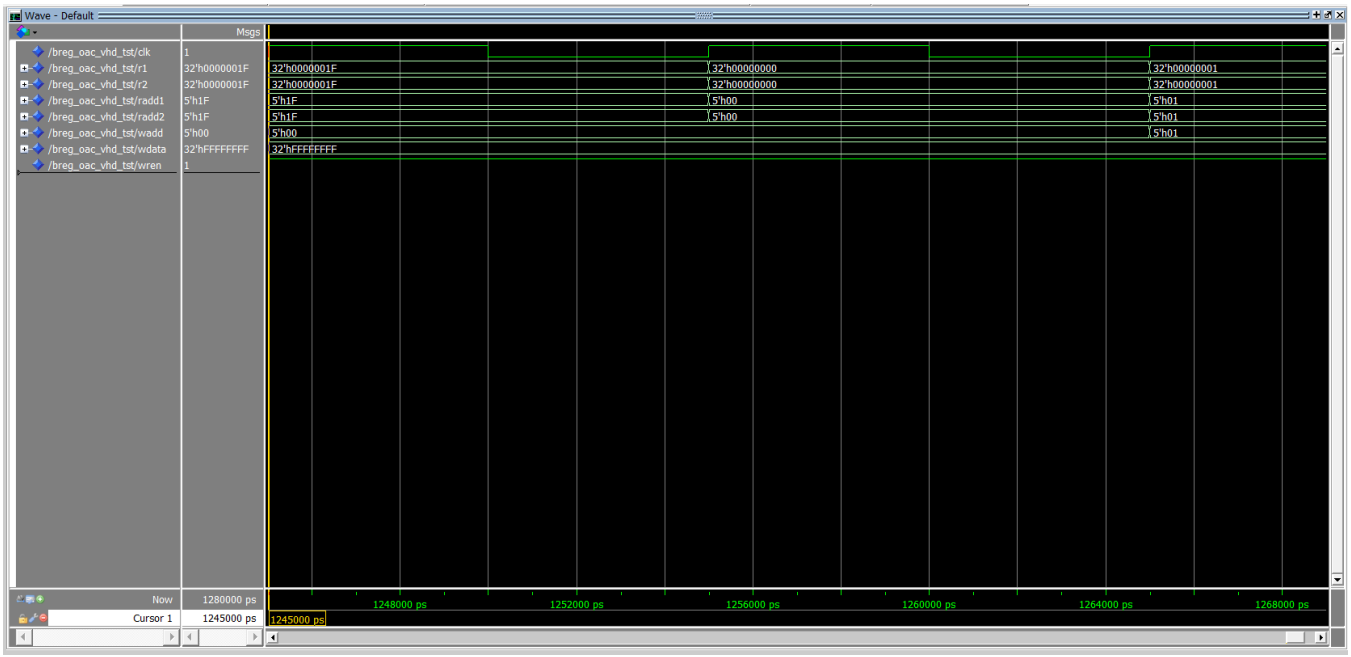

Figura 8: Tela do Caso 2 no *ModelSim*.

6.3 Caso 3

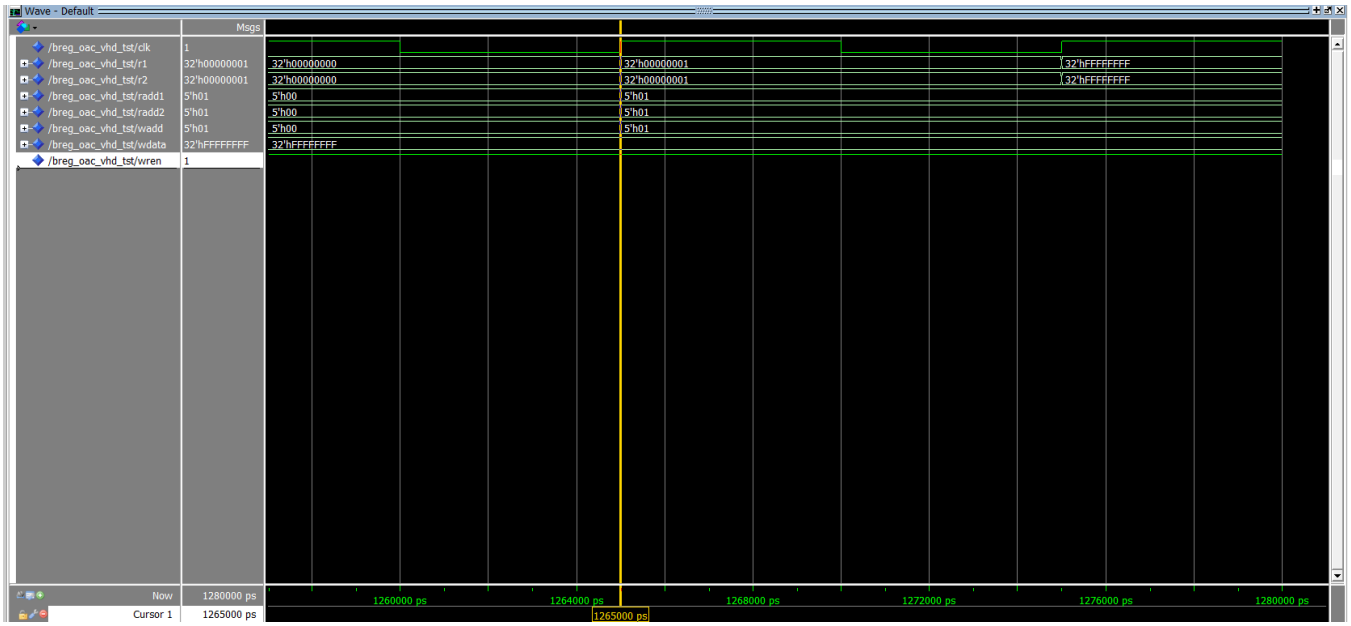

Figura 9: Tela do Caso 3 no *ModelSim*.

7 Dados da Síntese obtidos pelo *Quartus II*

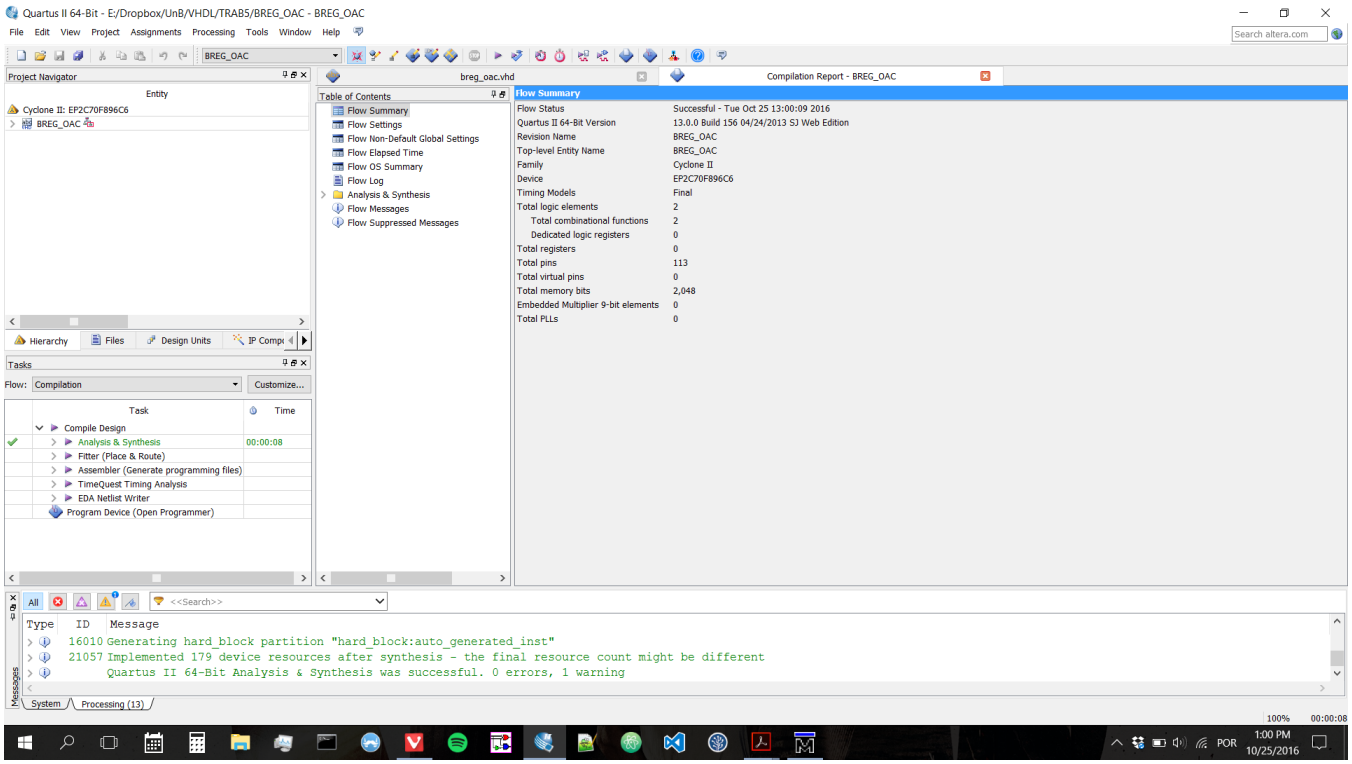

Figura 10: Dados da Sntese obtidos pelo *Quartus II* para a FPGA *Ciclone II EP270F896C6N*.