Relatório - Trabalho 5
Organização e Arquitetura de Computadores - Turma C

Gabriel Vieira de Arimatéa, 15/0126956

[1]CIC – Universidade de Brasília (UnB)

vieira.arimatea@gmail.com

1. Descrição do problema

O trabalho consite em projetar, simular e sintetizar um banco de registradores similar ao utilizado no MIPS.

2. Descrição sucinta do trabalho

O banco deveria possuir 32 registradores de 32 bits. Na borda de subida do clock, todas as entradas eram verificadas para saber o que se desejava. As portas de entrada são:

- **wren**: habilitação de escrita;
- **clk**: relógio do circuito;
- **rst**: sinal de reset, zera o conteúdo de todos os registradores do banco;
- **radd1**: endereço do registrador a ser lido em r1;
- **radd2**: endereço do registrador a ser lido em r2;
- **wdata**: valor a ser escrito;
- **wadd**: endereço do registrador a ser escrito o conteúdo de wdata.

As portas de saída são:

- **r1**: porta de saída para leitura do registrador endereçado por radd1;
- **r2**: porta de saída para leitura do registrador endereçado por radd2.

3. Códigos e resultados

3.1. Código do Banco de Registradores

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

entity bregMIPS is
   generic (WSIZE : natural := 32);
   port (
         clk, wren, rst        : 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 bregMIPS;

architecture Behavior of bregMIPS is
   type reg is array(WSIZE-1 downto 0) of std_logic_vector(WSIZE-1 downto 0);
   signal reg_mem: reg := (
         x"00000000", -- $ra
         x"00000000", -- $fp
         x"7fffeffc", -- $sp        Valor inicial no registrador de acordo com o MARS
         x"10008000", -- $gp        Valor inicial no registrador de acordo com o MARS
         x"00000000", -- $k1
         x"00000000", -- $k0
         x"00000000", -- $t9
         x"00000000", -- $t8
         x"00000000", -- $s7
         x"00000000", -- $s6
         x"00000000", -- $s5
         x"00000000", -- $s4
         x"00000000", -- $s3
         x"00000000", -- $s2
         x"00000000", -- $s1
         x"00000000", -- $s0
         x"00000000", -- $t7
         x"00000000", -- $t6
         x"00000000", -- $t5
         x"00000000", -- $t4
         x"00000000", -- $t3
         x"00000000", -- $t2
         x"00000000", -- $t1
         x"00000000", -- $t0
         x"00000000", -- $a3
         x"00000000", -- $a2
         x"00000000", -- $a1
         x"00000000", -- $a0
         x"00000000", -- $v1
         x"00000000", -- $v0
         x"00000000", -- $at
         x"00000000"  -- $zero
    );


   begin
   r1 <= reg_mem(to_integer(unsigned(radd1)));
   r2 <= reg_mem(to_integer(unsigned(radd2)));

   breg_mips: process(clk)
      begin

      if(rising_edge(clk)) then
         if(rst = '1') then
            reg_mem <= (
               x"00000000", -- $ra
               x"00000000", -- $fp
               x"7fffeffc", -- $sp
               x"10008000", -- $gp
               x"00000000", -- $k1
               x"00000000", -- $k0
               x"00000000", -- $t9
               x"00000000", -- $t8
               x"00000000", -- $s7
               x"00000000", -- $s6
               x"00000000", -- $s5
               x"00000000", -- $s4
               x"00000000", -- $s3
               x"00000000", -- $s2
               x"00000000", -- $s1
               x"00000000", -- $s0
               x"00000000", -- $t7
               x"00000000", -- $t6
               x"00000000", -- $t5
               x"00000000", -- $t4
               x"00000000", -- $t3
               x"00000000", -- $t2
               x"00000000", -- $t1
               x"00000000", -- $t0
               x"00000000", -- $a3
               x"00000000", -- $a2
               x"00000000", -- $a1
               x"00000000", -- $a0
               x"00000000", -- $v1
               x"00000000", -- $v0
               x"00000000", -- $at
               x"00000000"  -- $zero
            );
         end if;
         if ((wren = '1') and (to_integer(unsigned(wadd)) /= 0)) then
            reg_mem(to_integer(unsigned(wadd))) <= wdata;
         end if;
      end if;
      end process;

end Behavior;
```

3.2. Código do Test Bench

```vhdl
-- 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 "10/17/2018 20:21:43"

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

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

ENTITY bregMIPS_vhd_tst  IS
END bregMIPS_vhd_tst;
ARCHITECTURE bregMIPS_arch OF bregMIPS_vhd_tst  IS
-- constants
-- signals
```

```vhdl
SIGNAL clk : STD_LOGIC;
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 rst : STD_LOGIC;
SIGNAL wadd : STD_LOGIC_VECTOR(4 DOWNTO 0);
SIGNAL wdata : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL wren : STD_LOGIC;
COMPONENT bregMIPS
   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);
   rst : IN STD_LOGIC;
   wadd : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
   wdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
   wren : IN STD_LOGIC
   );
END COMPONENT;
BEGIN
   i1 : bregMIPS
   PORT MAP (
-- list connections between master ports and signals
   clk => clk,
   r1 => r1,
   r2 => r2,
   radd1 => radd1,
   radd2 => radd2,
   rst => rst,
   wadd => wadd,
   wdata => wdata,
   wren => wren
   );
init : PROCESS
-- variable declarations
BEGIN
        -- code that executes only once

      -- TESTE 1 --

        -- Preenchenco os registros com valores para serem lidos:

        wren <= '1';

        for i in 1 to 27 loop
            clk <= '0';
            wait for 2 ns;
            wadd <= std_logic_vector(to_unsigned(i, 5));
            wdata <= std_logic_vector(to_unsigned(i, 32));
            clk <= '1';
            wait for 2 ns;
         end loop;

         -- Pula registradores $gp e $sp

         for i in 30 to 31 loop
            clk <= '0';
            wait for 2 ns;
            wadd <= std_logic_vector(to_unsigned(i, 5));
            wdata <= std_logic_vector(to_unsigned(i, 32));
            clk <= '1';
            wait for 2 ns;
         end loop;

         -- Lendo R1
```

```vhdl

wren <= '0';

for i in 1 to 10 loop
   clk <= '0';
   wait for 2 ns;
   radd1 <= std_logic_vector(to_unsigned(i, 5));
   clk <= '1';
   wait for 2 ns;
end loop;
radd1 <= "00000";

-- Lendo R2

for i in 11 to 20 loop
   clk <= '0';
   wait for 2 ns;
   radd2 <= std_logic_vector(to_unsigned(i, 5));
   clk <= '1';
   wait for 2 ns;
end loop;

-- Lendo R1 e R2 simultaneamente

for i in 21 to 31 loop
   clk <= '0';
   wait for 2 ns;
   radd1 <= std_logic_vector(to_unsigned(i, 5));
   radd2 <= std_logic_vector(to_unsigned(i, 5));
```

```
            clk <= '1';
            wait for 2 ns;
         end loop;

      -- TESTE 2 --

         -- Verificando se há alguma alteração em $zero

         wren <= '1';

         clk <= '0';
         wait for 2 ns;

         wdata <= x"12345678";
         wadd <= "00000";

         clk <= '1';
         wait for 2 ns;

         wren <= '0';

         clk <= '0';
         wait for 2 ns;

         radd1 <= "00000";
         radd2 <= "00000";

         clk <= '1';
         wait for 2 ns;


      -- TESTE 3 --

         -- Restaurando os valores iniciais dos registradores

         rst <= '1';

```

```vhdl
        clk <= '0';
        wait for 2 ns;
        clk <= '1';
        wait for 2 ns;

        -- Apresentando valores iniciais

        for i in 0 to 31 loop
           clk <= '0';
           wait for 2 ns;
           radd1 <= std_logic_vector(to_unsigned(i, 5));
           clk <= '1';
           wait for 2 ns;
        end loop;

     -- TESTE 4 --

        -- Escrita e leitura no mesmo ciclo, do mesmo registrador

        wren <= '1';

        clk <= '0';
        wait for 2 ns;

        wdata <= x"0000000A";
        wadd <= "00001";

        radd1 <= "00001";
        radd2 <= "00001";

        clk <= '1';
        wait for 2 ns;

        -- Colocado mais um ciclo

        clk <= '0';
        wait for 2 ns;
        clk <= '1';
        wait for 2 ns;

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 bregMIPS_arch;
```

3.3. Resultado simulação (ModelSim)

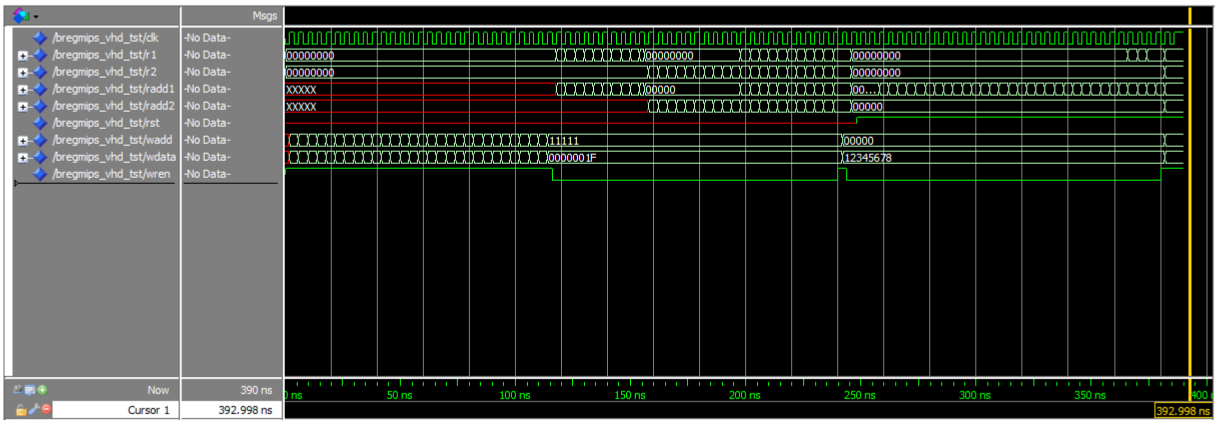

Figura 1. Resultado final total

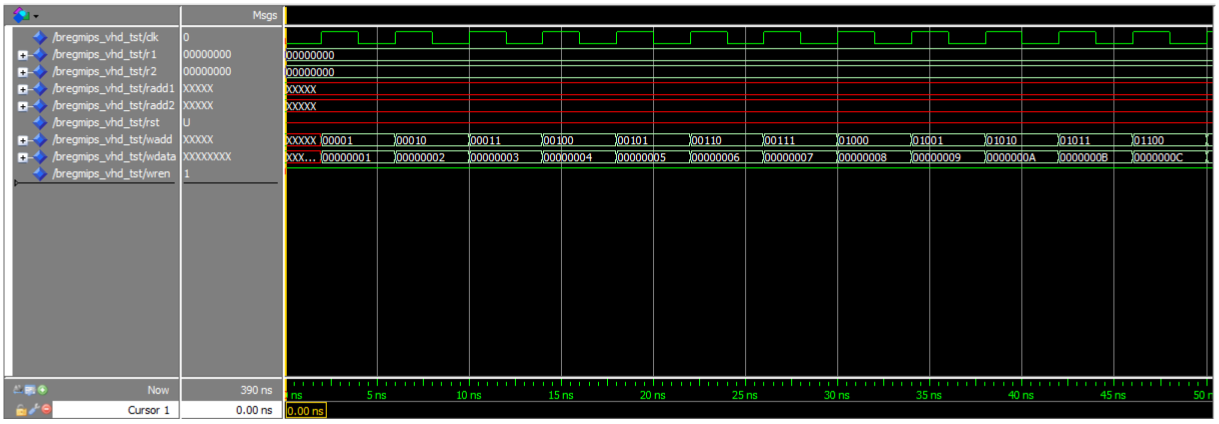

Figura 2. Primeira parte do Teste 1

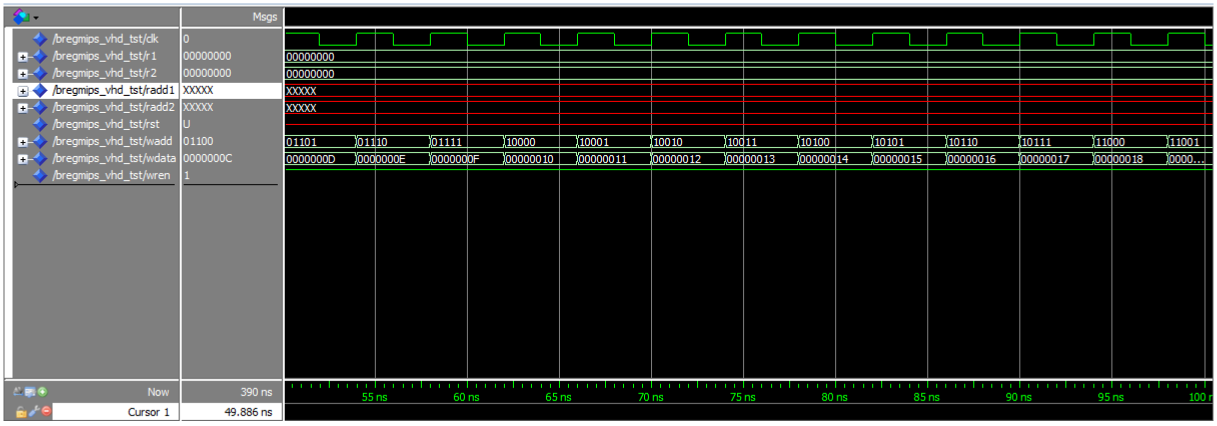
Msgs
/bregmips_vhd_tst/clk 0
/bregmips_vhd_tst/r1 00000000
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 XXXXX
/bregmips_vhd_tst/radd2 XXXXX
/bregmips_vhd_tst/rst U
/bregmips_vhd_tst/wadd 01100
/bregmips_vhd_tst/wdata 0000000C
/bregmips_vhd_tst/wren 1
01101 01110 01111 10000 10001 10010 10011 10100 10101 10110 10111 11000 11001
0000000D 0000000E 0000000F 00000010 00000011 00000012 00000013 00000014 00000015 00000016 00000017 00000018 0000...
Now 390 ns
Cursor 1 49.886 ns
55 ns 60 ns 65 ns 70 ns 75 ns 80 ns 85 ns 90 ns 95 ns 100 ns

Figura 3. Segunda parte do Teste 1

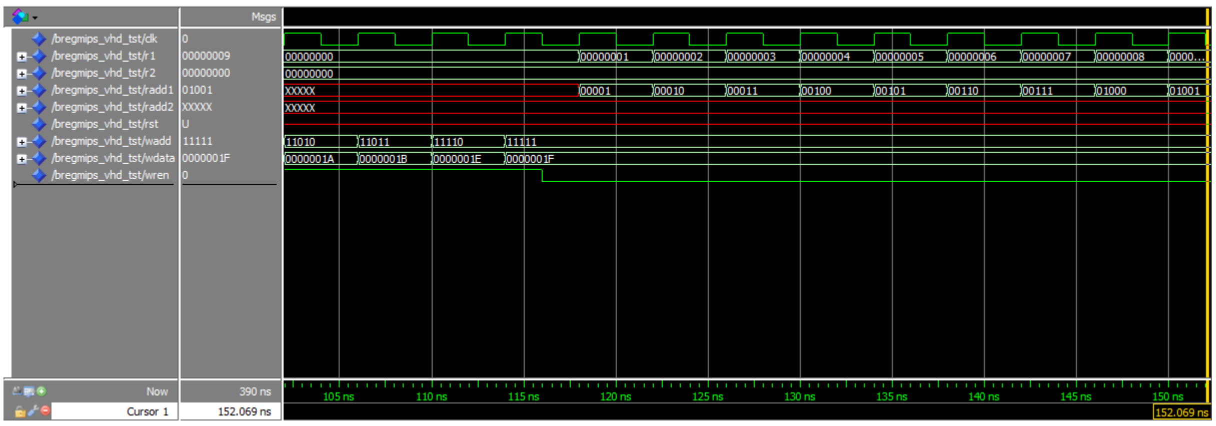
Msgs
/bregmips_vhd_tst/clk 0
/bregmips_vhd_tst/r1 00000009
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 01001
/bregmips_vhd_tst/radd2 XXXXX
/bregmips_vhd_tst/rst U
/bregmips_vhd_tst/wadd 11111
/bregmips_vhd_tst/wdata 0000001F
/bregmips_vhd_tst/wren 0
00000000 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 0000...
XXXXX 00001 00010 00011 00100 00101 00110 00111 01000 01001
11010 11011 11110 11111
0000001A 0000001B 0000001E 0000001F
Now 390 ns
Cursor 1 152.069 ns
105 ns 110 ns 115 ns 120 ns 125 ns 130 ns 135 ns 140 ns 145 ns 150 ns
152.069 ns

Figura 4. Terceira parte do Teste 1

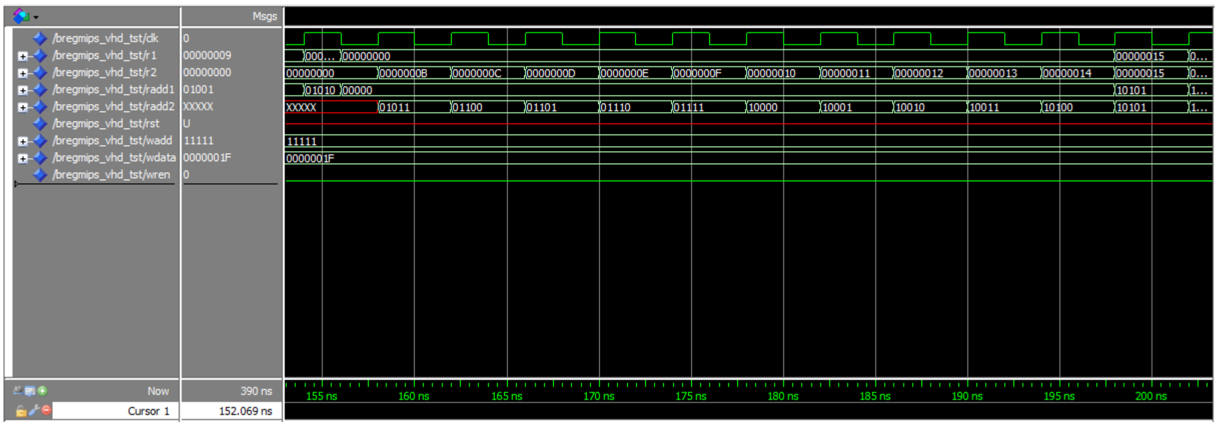
Msgs
/bregmips_vhd_tst/clk 0
/bregmips_vhd_tst/r1 00000009
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 01001
/bregmips_vhd_tst/radd2 XXXXX
/bregmips_vhd_tst/rst U
/bregmips_vhd_tst/wadd 11111
/bregmips_vhd_tst/wdata 0000001F
/bregmips_vhd_tst/wren 0
000... 00000000 00000015 0...
00000000 0000000B 0000000C 0000000D 0000000E 0000000F 00000010 00000011 00000012 00000013 00000014 00000015 0...
01010 00000 10101 1...
XXXXX 01011 01100 01101 01110 01111 10000 10001 10010 10011 10100 10101 1...
11111
0000001F
Now 390 ns
Cursor 1 152.069 ns
155 ns 160 ns 165 ns 170 ns 175 ns 180 ns 185 ns 190 ns 195 ns 200 ns

Figura 5. Quarta parte do Teste 1

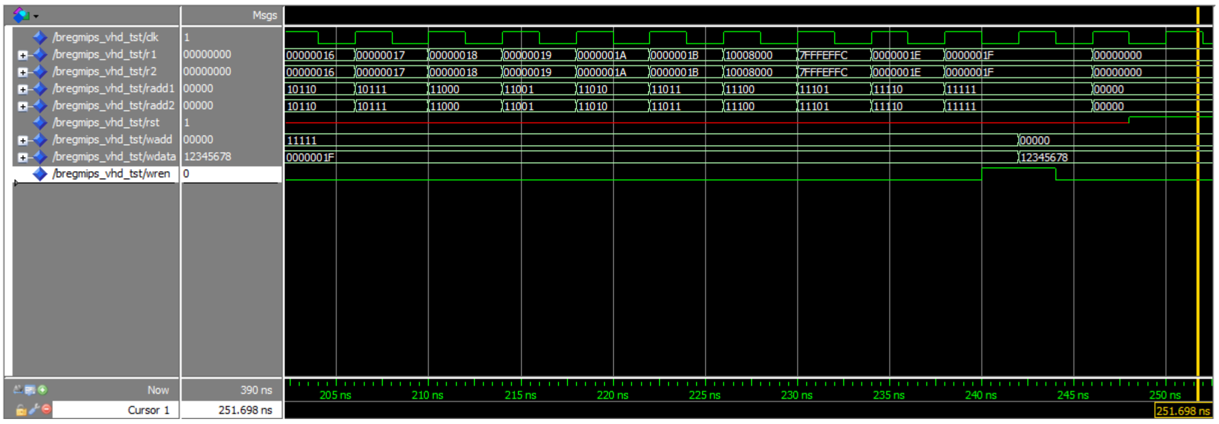
Msgs
/bregmips_vhd_tst/clk 1
/bregmips_vhd_tst/r1 00000000
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 00000
/bregmips_vhd_tst/radd2 00000
/bregmips_vhd_tst/rst 1
/bregmips_vhd_tst/wadd 00000
/bregmips_vhd_tst/wdata 12345678
/bregmips_vhd_tst/wren 0
00000016 00000017 00000018 00000019 0000001A 0000001B 10008000 7FFFEFFC 0000001E 0000001F 00000000
10110 10111 11000 11001 11010 11011 11100 11101 11110 11111 00000
11111 00000
0000001F 12345678
Now 390 ns
Cursor 1 251.698 ns
205 ns 210 ns 215 ns 220 ns 225 ns 230 ns 235 ns 240 ns 245 ns 250 ns
251.698 ns

Figura 6. Quinta parte do Teste 1 e Teste 2

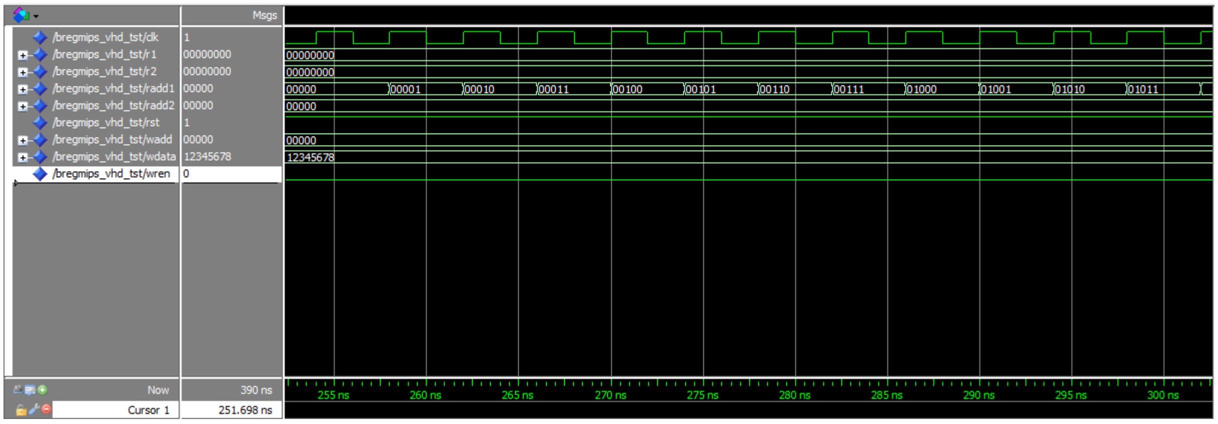
Msgs
/bregmips_vhd_tst/clk 1
/bregmips_vhd_tst/r1 00000000
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 00000
/bregmips_vhd_tst/radd2 00000
/bregmips_vhd_tst/rst 1
/bregmips_vhd_tst/wadd 00000
/bregmips_vhd_tst/wdata 12345678
/bregmips_vhd_tst/wren 0
00000000
00000000
00000 00001 00010 00011 00100 00101 00110 00111 01000 01001 01010 01011
00000
00000
12345678
Now 390 ns
Cursor 1 251.698 ns
255 ns 260 ns 265 ns 270 ns 275 ns 280 ns 285 ns 290 ns 295 ns 300 ns

Figura 7. Primeira parte do Teste 3

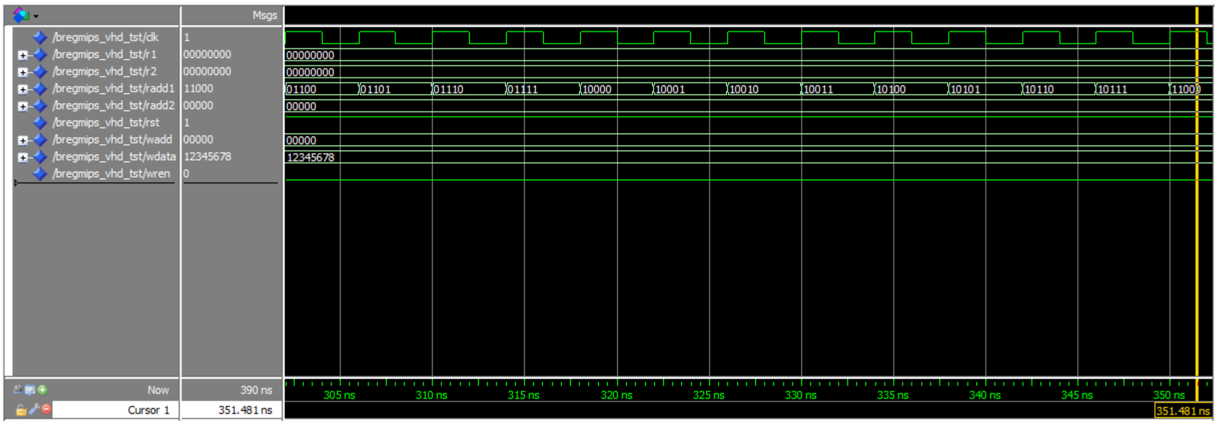
Msgs
/bregmips_vhd_tst/clk 1
/bregmips_vhd_tst/r1 00000000
/bregmips_vhd_tst/r2 00000000
/bregmips_vhd_tst/radd1 11000
/bregmips_vhd_tst/radd2 00000
/bregmips_vhd_tst/rst 1
/bregmips_vhd_tst/wadd 00000
/bregmips_vhd_tst/wdata 12345678
/bregmips_vhd_tst/wren 0
00000000
00000000
01100 01101 01110 01111 10000 10001 10010 10011 10100 10101 10110 10111 11000
00000
00000
12345678
Now 390 ns
Cursor 1 351.481 ns
305 ns 310 ns 315 ns 320 ns 325 ns 330 ns 335 ns 340 ns 345 ns 350 ns
351.481 ns

Figura 8. Segunda parte do Teste 3

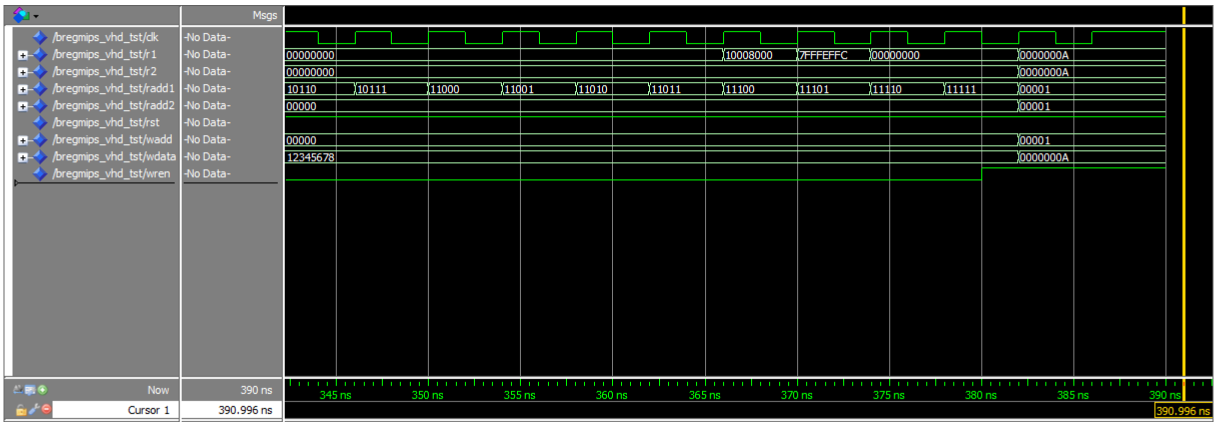
Msgs
/bregmips_vhd_tst/clk -No Data-
/bregmips_vhd_tst/r1 -No Data-
/bregmips_vhd_tst/r2 -No Data-
/bregmips_vhd_tst/radd1 -No Data-
/bregmips_vhd_tst/radd2 -No Data-
/bregmips_vhd_tst/rst -No Data-
/bregmips_vhd_tst/wadd -No Data-
/bregmips_vhd_tst/wdata -No Data-
/bregmips_vhd_tst/wren -No Data-
00000000
10008000
7FFFEFFC
00000000
0000000A
00000000
0000000A
10110
10111
11000
11001
11010
11011
11100
11101
11110
11111
00001
00000
00001
00000
00001
12345678
0000000A
345 ns
350 ns
355 ns
360 ns
365 ns
370 ns
375 ns
380 ns
385 ns
390 ns
Now 390 ns
Cursor 1 390.996 ns
390.996 ns

Figura 9. Terceira parte do Teste 3 e Teste 4

4. Conclusão

Como verificado, o registrador 0 ($zero) não pode ser auterado quando se escreve no mesmo.

Quando há a leitura e a escrita no mesmo registro e no mesmo período de clock, a saída recebe o valor atual do registrador.