Relatório - Trabalho 4
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 uma versão da ULA do MIPS de 32 bits no ambiente Quartus / ModelSim-Altera.

2. Descrição sucinta do trabalho

A ULA é uma peça fundamental da arquitetura de um processador. É nela que as operações são interpretadas e realizadas. Para este trabalho, foi necessário criar uma ULA utilizando linguagem VHDL. Suas entradas e saídas são:

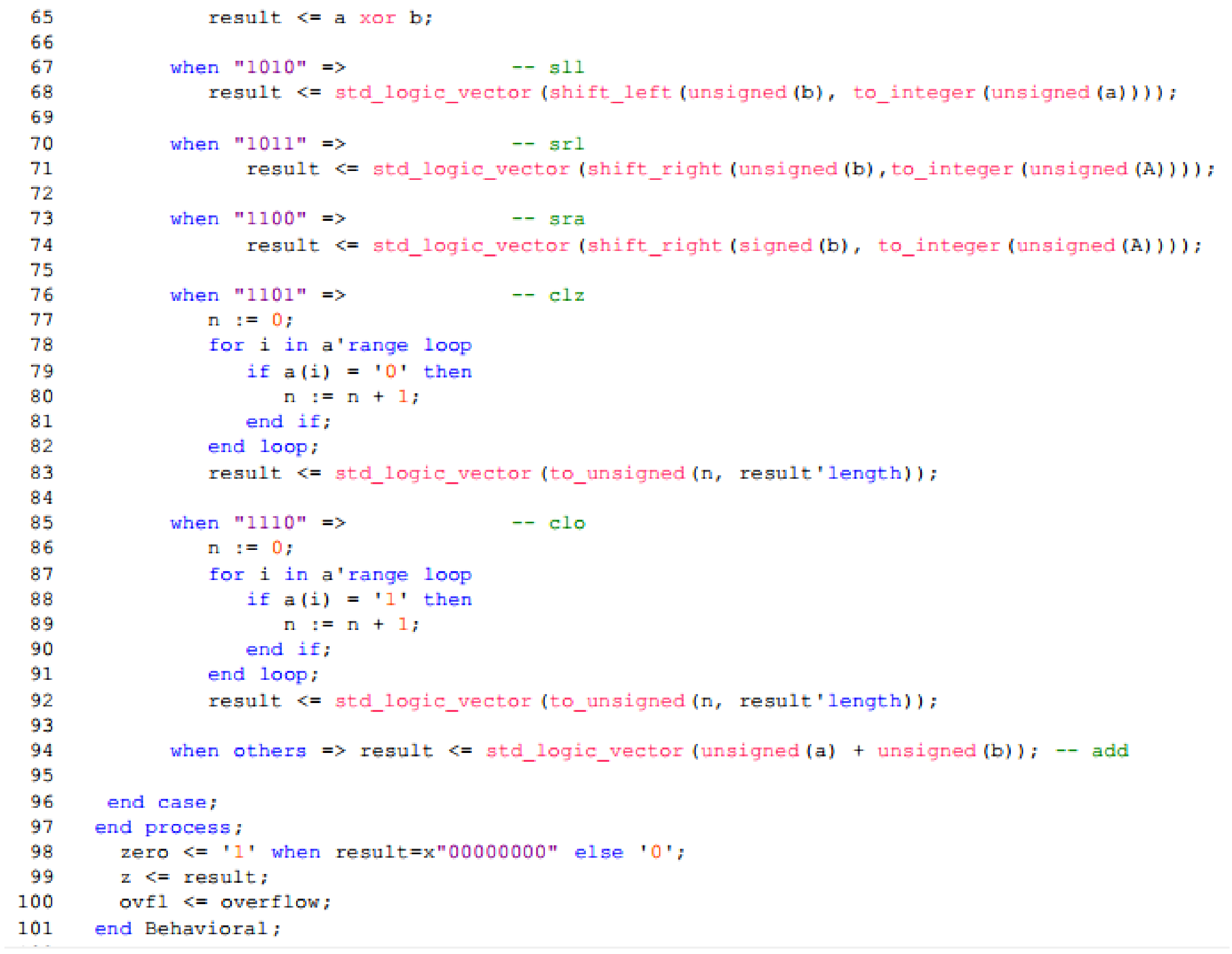
- **A** e **B**: entradas de dados;
- **Z**: saída de dados;
- **Zero**: detecta valor zero na saída;
- **Overflow**: ativo quando a operação de soma ou subtração gerar resultado que ultrapasse o limite de representação em 32 bits;
- **Opcode**: entrada com o opcode da instrução. Determina a operação que deve ser realziada.

3. Códigos e resultados

3.1. Código da ULA

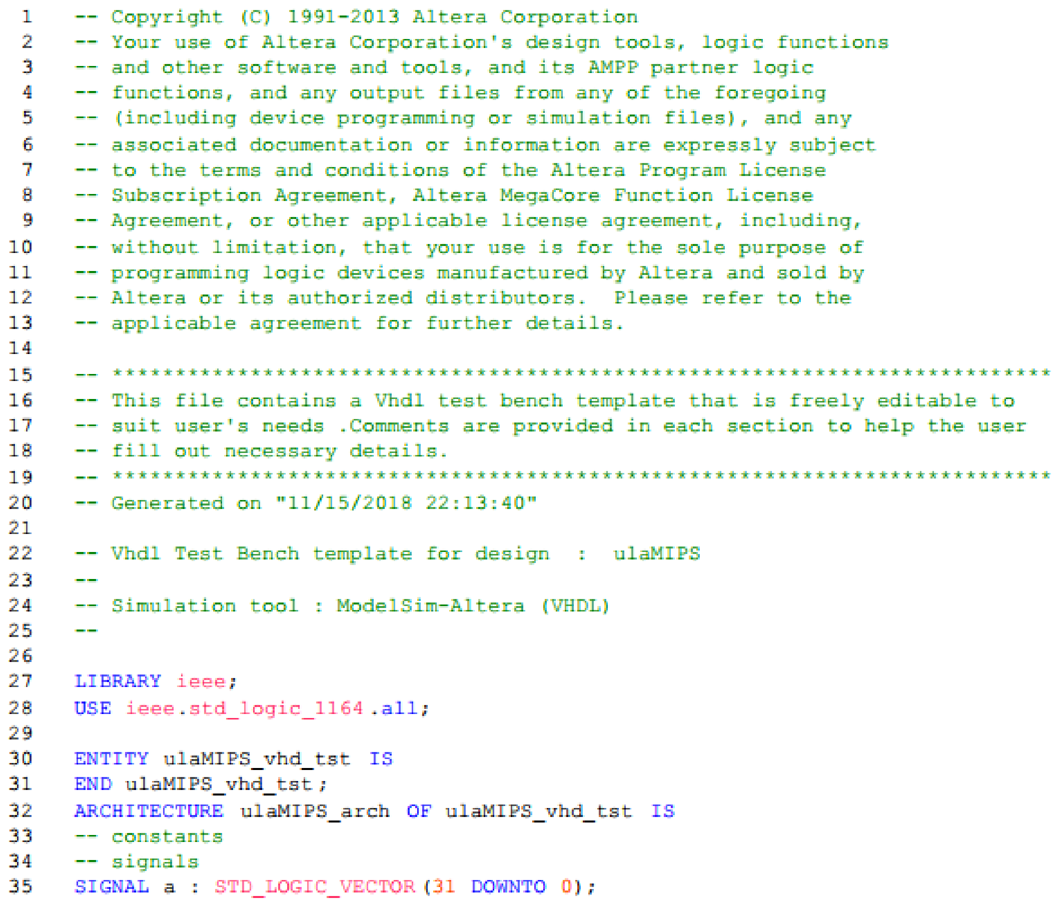
```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ulaMIPS is
   generic (WSIZE       : natural := 32);
      port (
            a, b           : in std_logic_vector  (WSIZE-1 downto 0);
            opcode         : in std_logic_vector(3 downto 0);
            z              : out std_logic_vector  (WSIZE-1 downto 0);
            zero, ovfl     : out std_logic
            );
end ulaMIPS;

architecture Behavioral of ulaMIPS is
   signal result: std_logic_vector(WSIZE-1 downto 0);
   signal overflow: std_logic;


   begin
   ula_mips: process(opcode,a,b)
   variable n : integer;
   begin
   overflow <= '0';
      case opcode is
      when "0000" =>          -- and
         result <= a and b;

      when "0001" =>          -- or
         result <= a or b;

      when "0010" =>              -- add
         result <= std_logic_vector(unsigned(a) + unsigned(b));
         overflow <= (a(WSIZE-1) and b(WSIZE-1) and not(result(WSIZE-1))) or (not(a(WSIZE
```

```vhdl
-1)) and not(b(WSIZE-1)) and result(WSIZE-1));

        when "0011" =>              -- addu
           result <= std_logic_vector(unsigned(a)+ unsigned(b));

        when "0100" =>              -- sub
           result <= std_logic_vector(unsigned(a) - unsigned(b));
           overflow <= (a(WSIZE-1) and b(WSIZE-1) and not(result(WSIZE-1))) or (not(a(WSIZE
-1)) and not(b(WSIZE-1)) and result(WSIZE-1));

        when "0101" =>              -- subu
           result <= std_logic_vector(unsigned(a) - unsigned(b));

        when "0110" =>           -- slt
           if (a<b) then
              result <= x"00000001";
           else
              result <= x"00000000";
           end if;

        when "0111" =>              -- sltu
           if (unsigned(a)<unsigned(b)) then
              result <= x"00000001";
           else
              result <= x"00000000";
           end if;

        when "1000" =>           -- nor
           result <= a nor b;

        when "1001" =>           -- xor
```

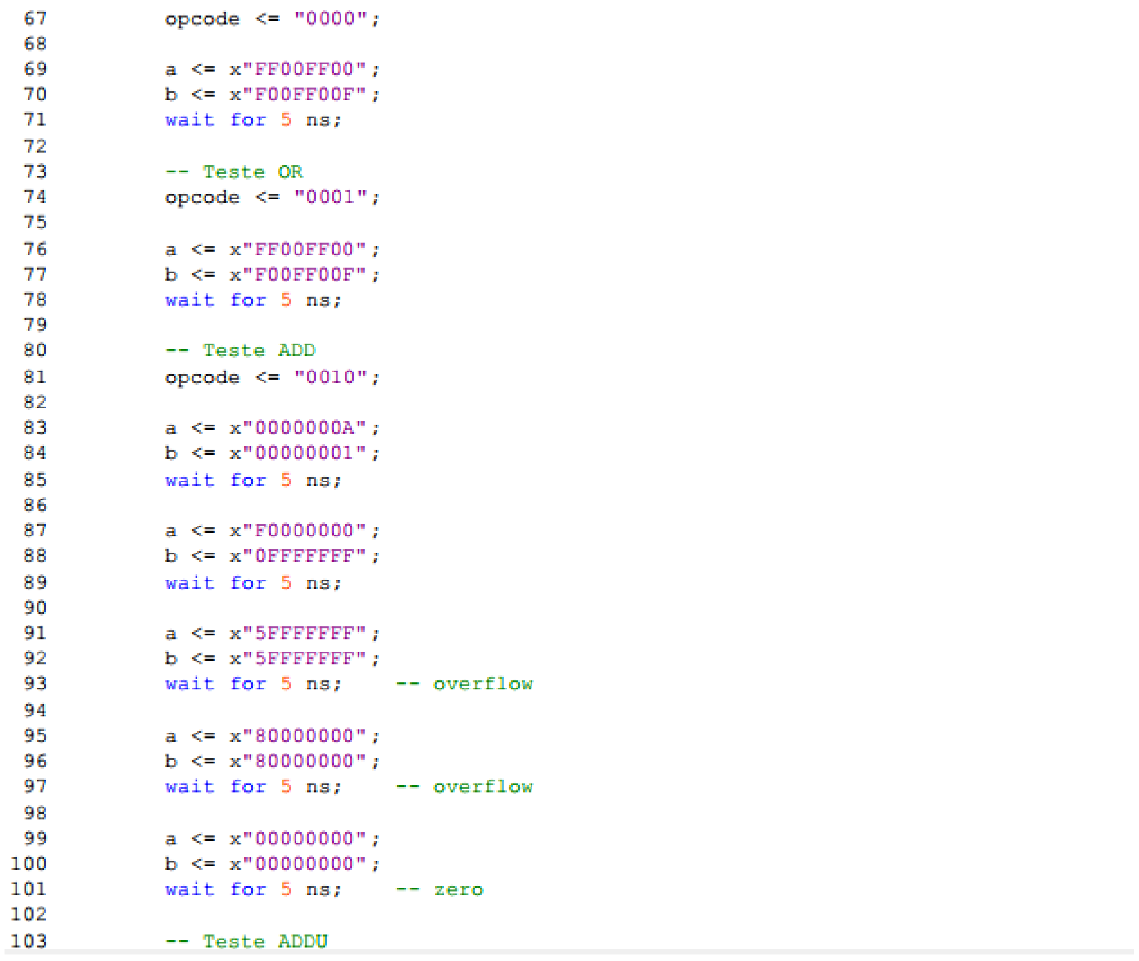


```vhdl
         result <= a xor b;

      when "1010" =>                -- sll
         result <= std_logic_vector(shift_left(unsigned(b), to_integer(unsigned(a))));

      when "1011" =>                -- srl
            result <= std_logic_vector(shift_right(unsigned(b),to_integer(unsigned(A))));

      when "1100" =>                -- sra
            result <= std_logic_vector(shift_right(signed(b), to_integer(unsigned(A))));

      when "1101" =>                -- clz
         n := 0;
         for i in a'range loop
            if a(i) = '0' then
               n := n + 1;
            end if;
         end loop;
         result <= std_logic_vector(to_unsigned(n, result'length));

      when "1110" =>                -- clo
         n := 0;
         for i in a'range loop
            if a(i) = '1' then
               n := n + 1;
            end if;
         end loop;
         result <= std_logic_vector(to_unsigned(n, result'length));

      when others => result <= std_logic_vector(unsigned(a) + unsigned(b)); -- add

 end case;
end process;
  zero <= '1' when result=x"00000000" else '0';
  z <= result;
  ovfl <= overflow;
end Behavioral;
```

3.2. Código do Test Bench

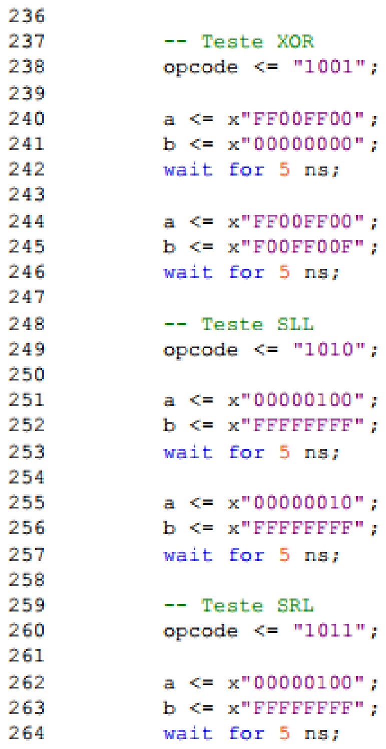


```vhdl
-- Copyright (C) 1991-2013 Altera Corporation
-- Your use of Altera Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- 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 "11/15/2018 22:13:40"

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

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ulaMIPS_vhd_tst IS
END ulaMIPS_vhd_tst;
ARCHITECTURE ulaMIPS_arch OF ulaMIPS_vhd_tst IS
-- constants
-- signals
SIGNAL a : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL b : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL opcode : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL ovfl : STD_LOGIC;
SIGNAL z : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL zero : STD_LOGIC;
COMPONENT ulaMIPS
   PORT (
   a : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
   b : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
   opcode : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
   ovfl : OUT STD_LOGIC;
   z : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
   zero : OUT STD_LOGIC
   );
END COMPONENT;
BEGIN
   i1 : ulaMIPS
   PORT MAP (
-- list connections between master ports and signals
   a => a,
   b => b,
   opcode => opcode,
   ovfl => ovfl,
   z => z,
   zero => zero
   );
init : PROCESS
-- variable declarations
BEGIN
        -- code that executes only once
      -- Teste AND
```

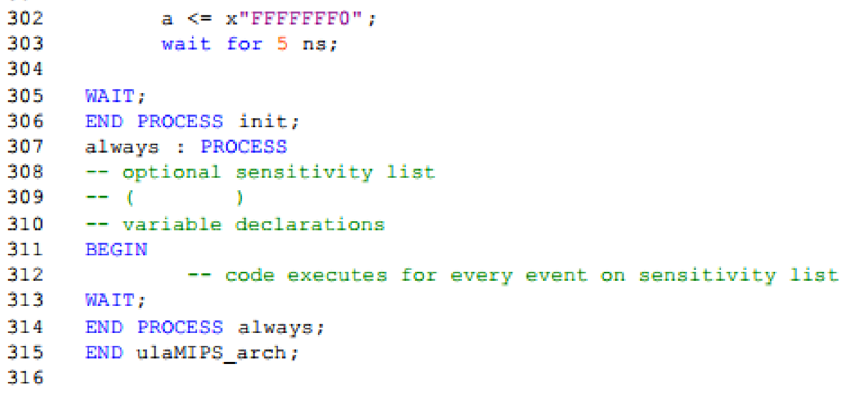


```vhdl
        opcode <= "0000";

        a <= x"FF00FF00";
        b <= x"F00FF00F";
        wait for 5 ns;

        -- Teste OR
        opcode <= "0001";

        a <= x"FF00FF00";
        b <= x"F00FF00F";
        wait for 5 ns;

        -- Teste ADD
        opcode <= "0010";

        a <= x"0000000A";
        b <= x"00000001";
        wait for 5 ns;

        a <= x"F0000000";
        b <= x"0FFFFFFF";
        wait for 5 ns;

        a <= x"5FFFFFFF";
        b <= x"5FFFFFFF";
        wait for 5 ns;    -- overflow

        a <= x"80000000";
        b <= x"80000000";
        wait for 5 ns;    -- overflow

        a <= x"00000000";
        b <= x"00000000";
        wait for 5 ns;    -- zero

        -- Teste ADDU
        opcode <= "0011";

        a <= x"0000000A";
        b <= x"00000001";
        wait for 5 ns;

        a <= x"F0000000";
        b <= x"0FFFFFFF";
        wait for 5 ns;

        a <= x"5FFFFFFF";
        b <= x"5FFFFFFF";
        wait for 5 ns;    -- overflow

        a <= x"80000000";
        b <= x"80000000";
        wait for 5 ns;    -- overflow

        a <= x"00000000";
        b <= x"00000000";
        wait for 5 ns;    -- zero

        -- Teste SUB
        opcode <= "0100";

        a <= x"00000000";
        b <= x"A0000000";
        wait for 5 ns;

        a <= x"A0000000";
        b <= x"60000000";
        wait for 5 ns;

        a <= x"80000000";
        b <= x"80000000";
        wait for 5 ns;    -- overflow

        a <= x"5FFFFFFF";
        b <= x"5FFFFFFF";
        wait for 5 ns;    -- overflow

        a <= x"A0000000";
        b <= x"A0000000";
        wait for 5 ns;    --zero

        -- Teste SUBU
        opcode <= "0101";

        a <= x"00000000";
        b <= x"A0000000";
        wait for 5 ns;

        a <= x"A0000000";
        b <= x"60000000";
        wait for 5 ns;

        a <= x"80000000";
        b <= x"80000000";
        wait for 5 ns;    -- overflow

        a <= x"5FFFFFFF";
        b <= x"5FFFFFFF";
        wait for 5 ns;    -- overflow

        a <= x"A0000000";
        b <= x"A0000000";
        wait for 5 ns;     --zero

        -- teste SLT
        opcode <= "0110";

        a <= x"000000C0";
        b <= x"000000A0";
        wait for 5 ns;

        a <= x"000000A0";
        b <= x"000000C0";
        wait for 5 ns;

        a <= x"000000C0";
        b <= x"000000C0";
        wait for 5 ns;

        a <= x"F00000C0";
        b <= x"000000A0";
        wait for 5 ns;

        a <= x"000000C0";
        b <= x"F00000A0";
        wait for 5 ns;

        a <= x"F00000C0";
        b <= x"F00000A0";
        wait for 5 ns;

```

```
        -- teste SLTU
        opcode <= "0111";

        a <= x"000000C0";
        b <= x"000000A0";
        wait for 5 ns;

        a <= x"000000A0";
        b <= x"000000C0";
        wait for 5 ns;

        a <= x"000000C0";
        b <= x"000000C0";
        wait for 5 ns;

        a <= x"F00000C0";
        b <= x"000000A0";
        wait for 5 ns;

        a <= x"000000C0";
        b <= x"F00000A0";
        wait for 5 ns;

        a <= x"F00000C0";
        b <= x"F00000A0";
        wait for 5 ns;

        -- Teste NOR
        opcode <= "1000";

        a <= x"FF00FF00";
        b <= x"F00FF00F";
        wait for 5 ns;

        a <= x"00000000";
        b <= x"00000000";
        wait for 5 ns;
```



```vhdl

        -- Teste XOR
        opcode <= "1001";

        a <= x"FF00FF00";
        b <= x"00000000";
        wait for 5 ns;

        a <= x"FF00FF00";
        b <= x"F00FF00F";
        wait for 5 ns;

        -- Teste SLL
        opcode <= "1010";

        a <= x"00000100";
        b <= x"FFFFFFFF";
        wait for 5 ns;

        a <= x"00000010";
        b <= x"FFFFFFFF";
        wait for 5 ns;

        -- Teste SRL
        opcode <= "1011";

        a <= x"00000100";
        b <= x"FFFFFFFF";
        wait for 5 ns;
```



```vhdl
      a <= x"FFFFFFF0";
      wait for 5 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 ulaMIPS_arch;

```

3.3. Resultado simolação (ModelSim)

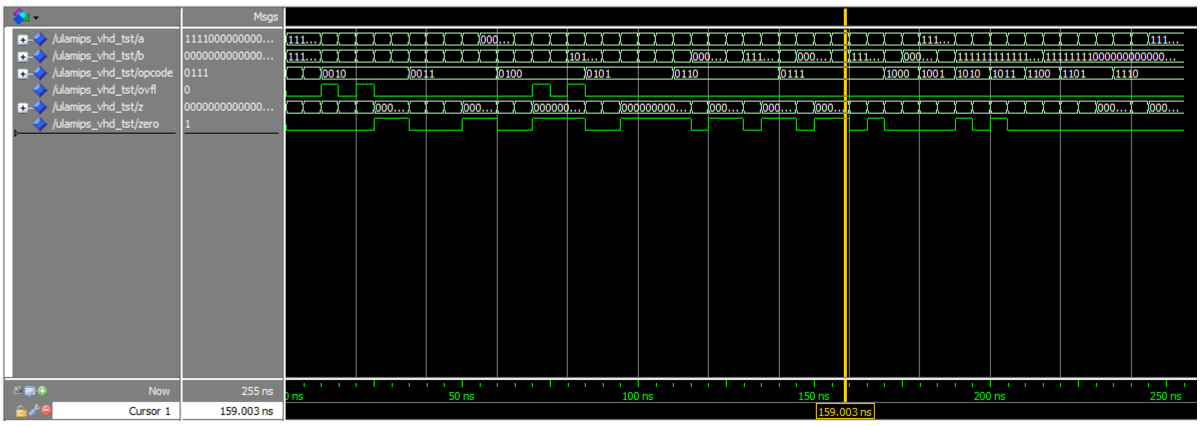

Msgs
/ulamips_vhd_tst/a 1111000000000...
/ulamips_vhd_tst/b 0000000000000...
/ulamips_vhd_tst/opcode 0111
/ulamips_vhd_tst/ovfl 0
/ulamips_vhd_tst/z 0000000000000...
/ulamips_vhd_tst/zero 1
Now 255 ns
Cursor 1 159.003 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
159.003 ns

Figura 1. Resultado final tota

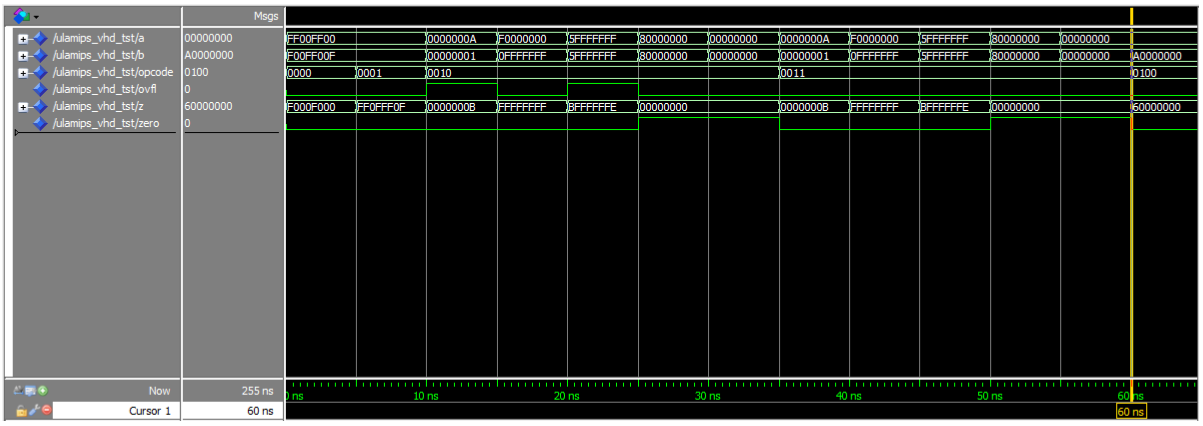


Figura 2. Parte 1

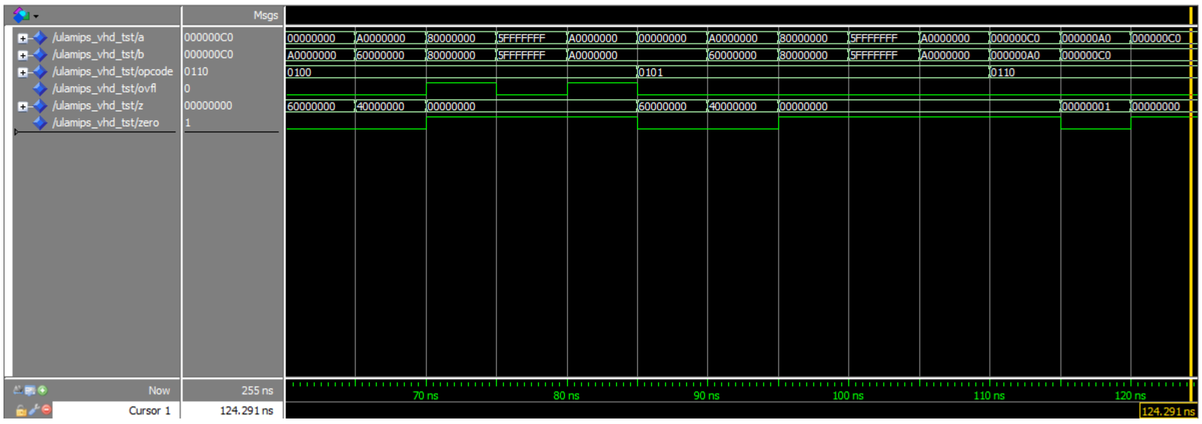


Figura 3. Parte 2

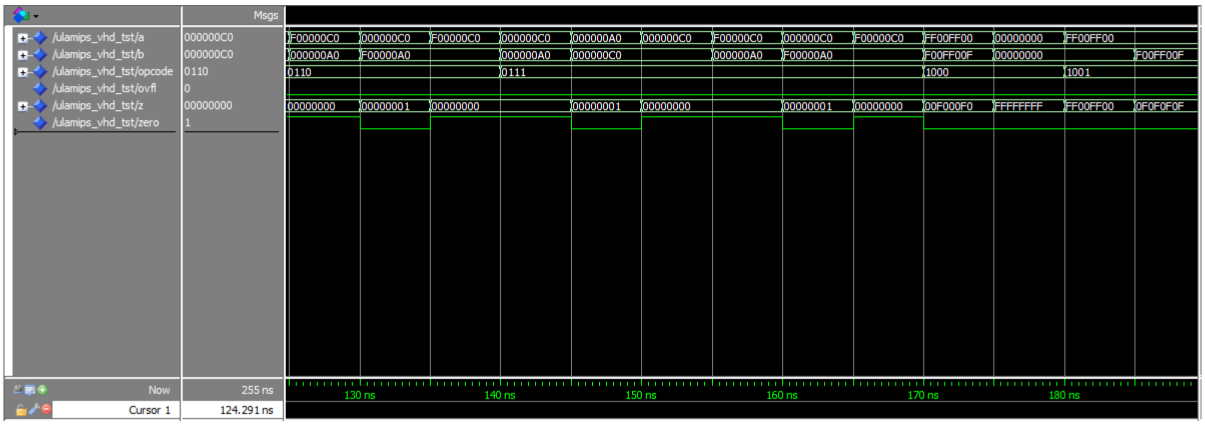


Figura 4. Parte 3

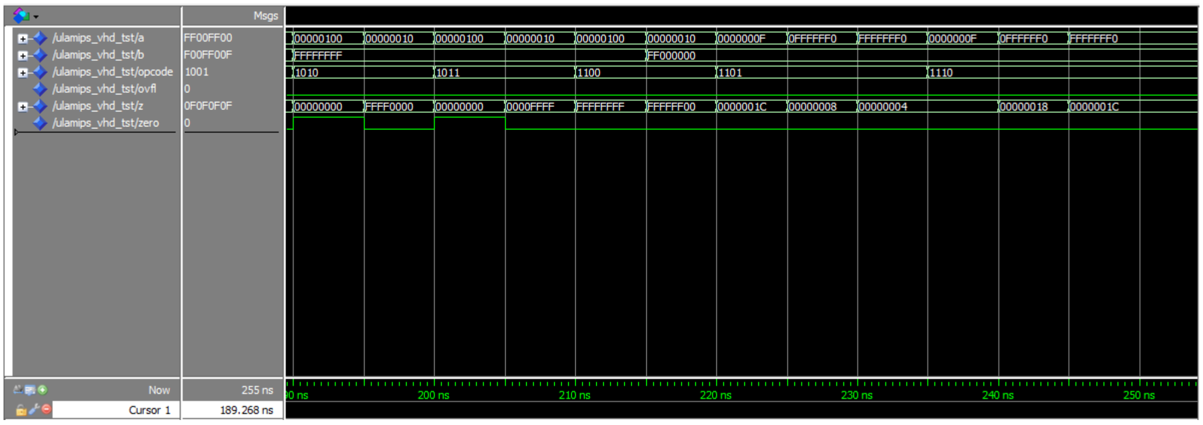


Figura 5. Parte 4

4. Observação

O trabalho foi entrege com uma semana de atraso com a permição do professor. Expliquei para ele que estava em uma competição e que não seria possível dedicar tempo para este trabalho na semana que era para o mesmo ser realizado.