

UNIVERSIDADE DE BRASÍLIA

INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES - TURMA C

Relatório

Trabalho IV: Projeto de uma ULA em VHDL

Aluno:

Lucas SANTOS - 14/0151010

Professor:

Ricardo JACOBI

23 de outubro de 2016



1 Objetivo

Projetar, simular e sintetizar uma versão da ULA do MIPS de 32 bits no ambiente *Quartus / ModelSim-Altera*.

2 Características da ULA

- Uma entrada *opcode*, que indica a operação a ser realizada;
- Duas entradas de dados: *A* e *B*;
- Uma Saída de dados: *Z*;
- Sinal Zero: detecta valor zero na saída *Z*;
- Sinal *Overflow*: ativo quando a operação de soma ou subtração gerar resultado que ultrapasse o limite de representação em 32 bits;
- Sinal Vai: indicação de vai-um no último bit da ULA;
- Operações realizadas em *Complemento de 2*.

3 Códigos

3.1 Código da ULA

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

-- Trabalho 4: Organizacao e Arquitetura de Computadores
-- Lucas Nascimento Santos Souza - 14/0151010

-- Projeto de uma ULA em VHDL
--      # Duas entradas de dados: A e B;
--      # Uma saida de dados: Z;
--      # Sinal Zero: detecta valor zero na saida;
--      # Sinal Overflow: ativo quando a operacao de soma ou subtracao gerar resultado
--      que ultrapasse o limite de representacao em 32 bits;
--      # Operacoes (em Complemento de 2).

-- Formato da ULA
entity ULA_OAC is
    generic (WSIZE : natural := 32);
    port(opcode : in std_logic_vector(3 downto 0);
          A, B : in std_logic_vector((WSIZE-1) downto 0);
          Z : out std_logic_vector((WSIZE-1) downto 0);
          vai, zero, ovfl : out std_logic );
end ULA_OAC;

architecture comportamento of ULA_OAC is
    -- A resposta obtida por determinada operacao
    signal resposta : std_logic_vector((WSIZE-1) downto 0);
    signal carry : std_logic;
    signal overflow : std_logic;

    -- Variaveis para implementacao do "vai"
    signal A33bits : std_logic_vector(WSIZE downto 0);
    signal B33bits : std_logic_vector(WSIZE downto 0);

    -- SOMA
    signal soma : std_logic_vector(WSIZE downto 0);

    -- SUBTRACAO
    signal subtracao : std_logic_vector(WSIZE downto 0);

    -- SHIFT
    signal numBitsToShift : natural;

begin
    -- SAIDAS
    -- Resultado da operacao em Z
    Z <= resposta;
    vai <= carry;
    ovfl <= overflow;

    -- A e B com 33 bits
    A33bits <= '0' & A;
    B33bits <= '0' & B;
```

```

-- SOMA
-- Resultado da soma c/ "vai"
soma <= std_logic_vector(unsigned(A33bits) + unsigned(B33bits));

-- SUBTRACAO
-- Resultado da subtracao c/ "vai"
subtracao <= std_logic_vector(unsigned(A33bits) - unsigned(B33bits));

-- SHIFT
-- Numero de bits que serao deslocados nos shifts
numBitsToShift <= (to_integer(unsigned(A)));

processo_ula: process(opcode, A, B, resposta, carry, overflow, soma, subtracao, numBitsToShift) begin

    -- Se Z tem valor 0, a saida "zero" eh ativa
    if (resposta = X"00000000") then zero <= '1'; else zero <= '0'; end if;

    -- Carry e Overflow se iniciam com valor 0
    carry <= '0';
    overflow <= '0';

    case opcode is
    -- AND
    when "0000" => resposta <= A and B;

    -- OR
    when "0001" => resposta <= A or B;

    -- ADD
    -- Soma c/ deteccao de overflow
    when "0010" => resposta <= std_logic_vector(unsigned(A) + unsigned(B)); -- A + B
    carry <= soma(WSIZE); -- carry_out = "vai"
    -- Se (+) + (+) = (-), Entao "ovfl" ativo
    -- Se (-) + (-) = (+), Entao "ovfl" ativo
    overflow <= (A(WSIZE-1) and B(WSIZE-1) and not(resposta(WSIZE-1)))
        or (not(A(WSIZE-1)) and not(B(WSIZE-1)) and resposta(WSIZE-1)); -- MSB: AB*Z + *A*BZ

    -- ADDU
    -- Soma s/ deteccao de overflow
    when "0011" => resposta <= std_logic_vector(unsigned(A) + unsigned(B)); -- A + B
    carry <= soma(WSIZE); -- carry_out = "vai"

    -- SUB
    -- Subtracao c/ deteccao de overflow
    when "0100" => resposta <= subtracao((WSIZE-1) downto 0); -- A - B s/ borrow_out
    carry <= subtracao(WSIZE); -- borrow_out = "vai"
    -- Se (+) - (-) = (-), Entao "ovfl" ativo
    -- Se (-) - (+) = (+), Entao "ovfl" ativo
    overflow <= (A(WSIZE-1) and not(B(WSIZE-1)) and not(resposta(WSIZE-1)))
        or (not(A(WSIZE-1)) and B(WSIZE-1) and resposta(WSIZE-1)); -- MSB: A*B*Z + *ABZ

    -- SUBU
    -- Subtracao s/ deteccao de overflow
    when "0101" => resposta <= subtracao((WSIZE-1) downto 0); -- A - B s/ borrow_out
    carry <= subtracao(WSIZE); -- borrow_out = "vai"

```

```

-- SLT
-- Se  $A < B$ , Entao resposta = 1
-- resposta(0) <= bit de sinal = subtracao(WSIZE-1)
when "0110" => resposta <= (0 => subtracao(WSIZE-1), others => '0');

-- NAND
when "0111" => resposta <= A nand B;

-- NOR
when "1000" => resposta <= A nor B;

-- XOR
when "1001" => resposta <= A xor B;

-- SLL
-- Shift Left Logical
-- Performs a shift-left on an UNSIGNED vector COUNT times.
-- The vacated positions are filled with '0'.
-- The COUNT leftmost elements are lost.
when "1010" => resposta <= std_logic_vector(shift_left(unsigned(B), numBitsToShift));

-- SRL
-- Shift Right Logical
-- Performs a shift-right on an UNSIGNED vector COUNT times.
-- The vacated positions are filled with '0'.
-- The COUNT rightmost elements are lost.
when "1011" => resposta <= std_logic_vector(shift_right(unsigned(B), numBitsToShift));

-- SRA
-- Shift Right Arithmetic
-- Performs a shift-right on a SIGNED vector COUNT times.
-- The vacated positions are filled with the leftmost element, ARG'LEFT.
-- The COUNT rightmost elements are lost.
when "1100" => resposta <= std_logic_vector(shift_right(signed(B), numBitsToShift));

-- Se outros opcodes, Entao resposta = 0
when others => resposta <= (others => '0');

end case;

end process;

end architecture comportamento;

```

3.2 Código do TestBench

```
-- Copyright (C) 1991-2011 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 "10/18/2016 21:03:45"

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

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ULA_OAC_vhd_tst IS
END ULA_OAC_vhd_tst;

ARCHITECTURE ULA_OAC_arch OF ULA_OAC_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 vai : STD_LOGIC;
    SIGNAL Z : STD_LOGIC_VECTOR(31 DOWNTO 0);
    SIGNAL zero : STD_LOGIC;

    COMPONENT ULA_OAC
        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;
            vai : OUT STD_LOGIC;
            Z : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
            zero : OUT STD_LOGIC
        );
    END COMPONENT;
END COMPONENT;
```

```

BEGIN
    i1 : ULA_OAC
    PORT MAP (
-- list connections between master ports and signals
        A => A,
        B => B,
        opcode => opcode,
        ovfl => ovfl,
        vai => vai,
        Z => Z,
        zero => zero
    );

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

-- TESTANDO AND
A <= X"FFFF0000"; B <= X"FF00FF00"; opcode <= "0000"; wait for 100 ns;
-- and = Z => 0xFFFF0000
A <= X"FFFF0000"; B <= X"0000FFFF"; opcode <= "0000"; wait for 100 ns;
-- and = Z => 0x0 , zero = 1

-- TESTANDO OR
A <= X"FFFF0000"; B <= X"FF00FF00"; opcode <= "0001"; wait for 100 ns;
-- or = Z => 0xFFFFFF00
A <= X"FFFF0000"; B <= X"0000FFFF"; opcode <= "0001"; wait for 100 ns;
-- or = Z => 0xFFFFFFFF

-- TESTANDO ADD
-- zero
A <= X"00000000"; B <= X"00000000"; opcode <= "0010"; wait for 100 ns;
-- add = Z => 0x0, zero = 1

-- negativo
A <= X"FFFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0010"; wait for 100 ns;
-- add = Z => 0xFFFFFFF0, carry = 1

-- positivo
A <= X"0000FFFF"; B <= X"0000FFFF"; opcode <= "0010"; wait for 100 ns;
-- add = Z => 0x1FFFE

-- overflow
-- P + P = N
A <= X"7FFFFFFF"; B <= X"7FFFFFFF"; opcode <= "0010"; wait for 100 ns;
-- add = Z => 0xFFFFFFF0, ovfl = 1, carry = 0

-- overflow
-- N + N = P
A <= X"80000000"; B <= X"80000000"; opcode <= "0010"; wait for 100 ns;
-- add = Z => 0x10000000, ovfl = 1, carry = 1

-- TESTANDO ADDU
-- zero
A <= X"00000000"; B <= X"00000000"; opcode <= "0011"; wait for 100 ns;

```

```

-- add = Z => 0x0, zero = 1

-- negativo
A <= X"FFFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0011"; wait for 100 ns;
-- add = Z => 0xFFFFFFFF, carry = 1

-- positivo
A <= X"0000FFFF"; B <= X"0000FFFF"; opcode <= "0011"; wait for 100 ns;
-- add = Z => 0x1FFFFE

-- overflow
-- P + P = N
A <= X"7FFFFFFF"; B <= X"7FFFFFFF"; opcode <= "0011"; wait for 100 ns;
-- addu = Z => 0xFFFFFFFF, ovfl = 0, carry = 0

-- overflow
-- N + N = P
A <= X"80000000"; B <= X"80000000"; opcode <= "0011"; wait for 100 ns;
-- addu = Z => 0x100000000, ovfl = 0, carry = 1

-- TESTANDO SUB
-- zero
A <= X"FFFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0100"; wait for 100 ns;
-- add = Z => 0x00000000, zero = 1

-- negativo
A <= X"00000000"; B <= X"00000001"; opcode <= "0100"; wait for 100 ns;
-- add = Z => 0xFFFFFFFF, ovfl = 1

-- positivo
A <= X"0000000A"; B <= X"00000006"; opcode <= "0100"; wait for 100 ns;
-- add = Z => 0x4

-- overflow
-- N - P = P
A <= X"80000000"; B <= X"00000001"; opcode <= "0100"; wait for 100 ns;
-- sub = 0x7FFFFFFF, ovfl = 1, borrow = 0

-- overflow
-- P - N = N
A <= X"7FFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0100"; wait for 100 ns;
-- sub = 0x80000000, ovfl = 1, borrow = 1

-- TESTANDO SUBU
-- zero
A <= X"FFFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0101"; wait for 100 ns;
-- add = Z => 0x0, zero = 1

-- negativo
A <= X"00000000"; B <= X"00000001"; opcode <= "0101"; wait for 100 ns;
-- add = Z => 0xFFFFFFFF, ovfl = 1

-- positivo
A <= X"0000000A"; B <= X"00000006"; opcode <= "0101"; wait for 100 ns;
-- add = Z => 0x4

```



```

-- overflow
-- N - P = P
A <= X"80000000"; B <= X"00000001"; opcode <= "0101"; wait for 100 ns;
-- subu = 0x7FFFFFFF, ovfl = 0, borrow = 0

-- overflow
-- P - N = N
A <= X"7FFFFFFF"; B <= X"FFFFFFFF"; opcode <= "0101"; wait for 100 ns;
-- subu = 0x80000000, ovfl = 0, borrow = 1

-- TESTANDO SET ON LESS THEN
A <= X"0000000A"; B <= X"0000000F"; opcode <= "0110"; wait for 100 ns;
-- slt = 0x1
A <= X"0000000A"; B <= X"00000006"; opcode <= "0110"; wait for 100 ns;
-- slt = 0x0, zero = 1

-- TESTANDO NAND
A <= X"000000FF"; B <= X"000000FF"; opcode <= "0111"; wait for 100 ns;
-- nand = 0xFFFFFFFF00
A <= X"FF0000FF"; B <= X"FF0000FF"; opcode <= "0111"; wait for 100 ns;
-- nand = 0x00FFFF00

-- TESTANDO NOR
A <= X"00000000"; B <= X"00000000"; opcode <= "1000"; wait for 100 ns;
-- nor = 0xFFFFFFFF
A <= X"0000FFFF"; B <= X"FF00FF00"; opcode <= "1000"; wait for 100 ns;
-- nor = 0x00FF0000

-- TESTANDO XOR
A <= X"00000000"; B <= X"00000001"; opcode <= "1001"; wait for 100 ns;
-- xor = 0x1
A <= X"00000010"; B <= X"00000001"; opcode <= "1001"; wait for 100 ns;
-- xor = 0x11

-- TESTANDO SHIFT LEFT LOGICAL
A <= X"00000010"; B <= X"FFFFFFFF"; opcode <= "1010"; wait for 100 ns;
-- sll = 0xFFFF0000
A <= X"00000100"; B <= X"FFFFFFFF"; opcode <= "1010"; wait for 100 ns;
-- sll = 0x0, zero = 1

-- TESTANDO SHIFT RIGHT LOGICAL
A <= X"00000010"; B <= X"FFFFFFFF"; opcode <= "1011"; wait for 100 ns;
-- srl = 0x0000FFFF
A <= X"00000001"; B <= X"FFFFFFFF"; opcode <= "1011"; wait for 100 ns;
-- srl = 0x7FFFFFFF

-- TESTANDO SHIFT RIGHT ARITHMETIC
A <= X"00000010"; B <= X"FFFFFFFF"; opcode <= "1100"; wait for 100 ns;
-- sra = 0xFFFFFFFF
A <= X"00000010"; B <= X"FF0000FF"; opcode <= "1100"; wait for 100 ns;
-- sra = 0xFFFFFFFF00

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 ULA_OAC_arch;
```

4 Simulação no *ModelSim*



Figura 1: Primeira tela de simulação do *ModelSim*.



Figura 2: Segunda tela de simulação do *ModelSim*.



Figura 3: Terceira tela de simulação do *ModelSim*.

5 Dados da Síntese obtidos pelo *Quartus II*

The screenshot displays the Quartus II 32-bit software interface. The top menu bar includes File, Edit, View, Project, Assignments, Processing, Tools, Window, and Help. The main workspace is divided into several panes:

- Project Navigator:** Shows the project hierarchy with 'Cyclone II: EP2C70F896C6' and 'ULA_OAC'.
- Table of Contents:** Lists the report sections: Flow Summary, Flow Settings, Flow Non-Default Global Settings, Flow Elapsed Time, Flow OS Summary, Flow Log, and Analysis & Synthesis.
- Flow Summary:** A detailed report of the synthesis process.
- Messages:** A log of system messages, including information about the entity 'ULA_OAC' and a warning about parallel compilation.

The **Flow Summary** report provides the following data:

| Flow Status | Successful - Thu Oct 20 16:29:49 2016 |
|------------------------------------|--|
| Quartus II 32-bit Version | 11.1 Build 173 11/01/2011 SJ Web Edition |
| Revision Name | ULA_OAC |
| Top-level Entity Name | ULA_OAC |
| Family | Cyclone II |
| Device | EP2C70F896C6 |
| Timing Models | Final |
| Total logic elements | 745 |
| Total combinational functions | 745 |
| Dedicated logic registers | 0 |
| Total registers | 0 |
| Total pins | 103 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

The Messages pane shows the following log entries:

- Info (12127): Elaborating entity "ULA_OAC" for the top level hierarchy
- Info (16010): Generating hard_block partition "hard_block:auto_generated_inst"
- Info (21057): Implemented 848 device resources after synthesis - the final resource count might be different
- Info: Command: quartus_map --read_settings_files=on --write_settings_files=off ULA_OAC -c ULA_OAC
- Info: Quartus II 32-bit Analysis & Synthesis was successful. 0 errors, 1 warning
- Info: Running Quartus II 32-bit Analysis & Synthesis
- Warning (20028): Parallel compilation is not licensed and has been disabled

The bottom status bar indicates 'Message: 0 of 20' and 'Location:'. The bottom right corner shows '100%' and '00:00:36'.

Figura 4: Dados da Síntese obtidos pelo *Quartus II* para a FPGA *Cyclone II EP270F896C6N*.