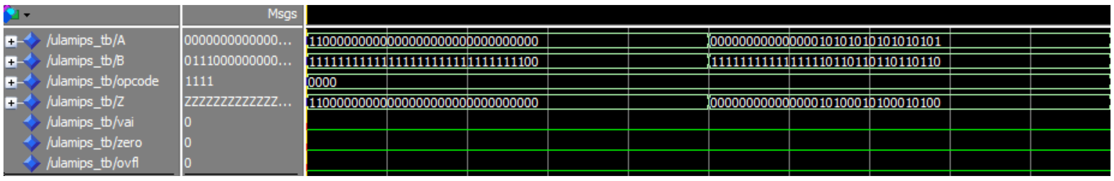
Trabalho 4 de Arquitetura e Organização de Arquivos – Turma C
Aluno: Marcos Paulo Cayres Rosa
Matrícula: 14/0027131

Projeto de uma ULA em VHDL

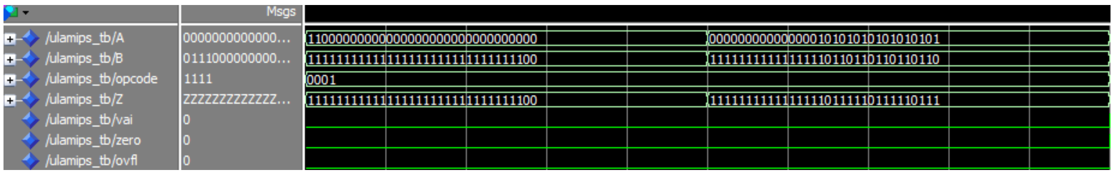
Telas de simulação:

Para as operações aritméticas, os testes são feitos para resultado e operando zero, casos com overflow e carry e utilizando valores positivos e negativos.

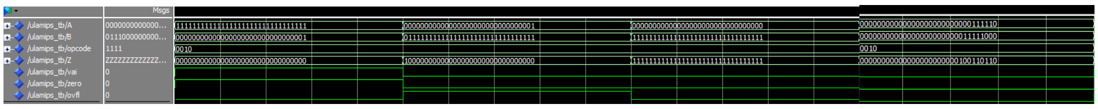
AND


```
/ulamips_tb/A        0000000000000...  11000000000000000000000000000000  00000000000000010101010101010101
/ulamips_tb/B        0111000000000...  11111111111111111111111111111100  11111111111111110110110110110110
/ulamips_tb/opcode   1111              0000
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  11000000000000000000000000000000  00000000000000010100010100010100
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

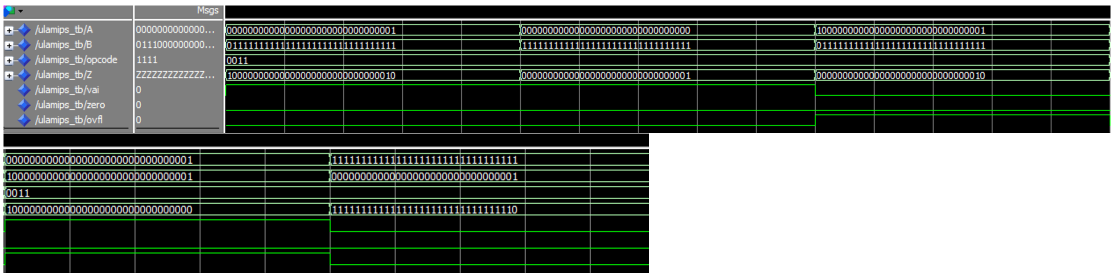
OR


```
/ulamips_tb/A        0000000000000...  11000000000000000000000000000000  00000000000000010101010101010101
/ulamips_tb/B        0111000000000...  11111111111111111111111111111100  11111111111111110110110110110110
/ulamips_tb/opcode   1111              0001
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  11111111111111111111111111111100  11111111111111110111110111110111
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

ADD

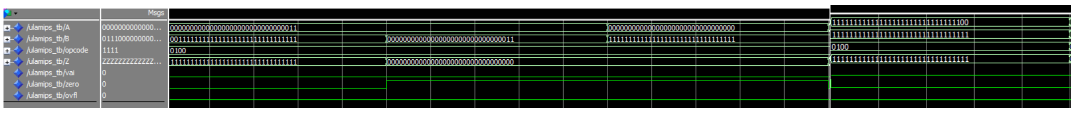

```
/ulamips_tb/A        0000000000000...  11111111111111111111111111111111  00000000000000000000000000000001  00000000000000000000000000000000  00000000000000000000000000111110
/ulamips_tb/B        0111000000000...  00000000000000000000000000000001  01111111111111111111111111111111  11111111111111111111111111111111  00000000000000000000000011111000
/ulamips_tb/opcode   1111              0010                                                                                                                   0010
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  00000000000000000000000000000000  10000000000000000000000000000000  11111111111111111111111111111111  00000000000000000000000100110110
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

SUB

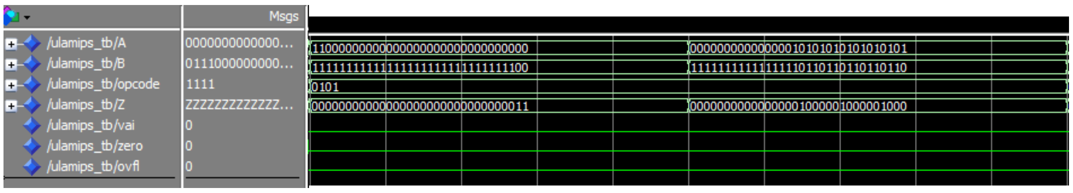

```
/ulamips_tb/A        0000000000000...  00000000000000000000000000000001  00000000000000000000000000000000  10000000000000000000000000000001
/ulamips_tb/B        0111000000000...  01111111111111111111111111111111  11111111111111111111111111111111  01111111111111111111111111111111
/ulamips_tb/opcode   1111              0011
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  10000000000000000000000000000010  00000000000000000000000000000001  00000000000000000000000000000010
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0

00000000000000000000000000000001  11111111111111111111111111111111
10000000000000000000000000000001  00000000000000000000000000000001
0011
10000000000000000000000000000000  11111111111111111111111111111110
```

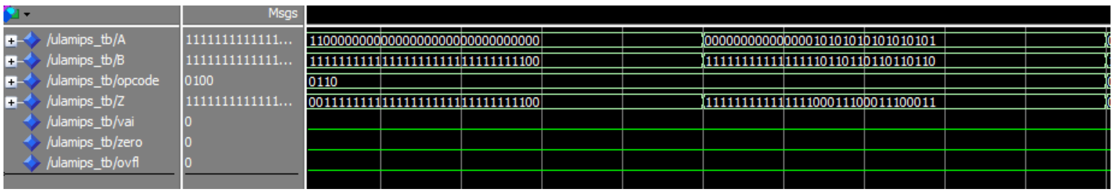
SLT


```
/ulamips_tb/A        0000000000000...  00000000000000000000000000000011                                    00000000000000000000000000000000  11111111111111111111111111111100
/ulamips_tb/B        0111000000000...  00111111111111111111111111111111  00000000000000000000000000000011  11111111111111111111111111111111  11111111111111111111111111111111
/ulamips_tb/opcode   1111              0100                                                                                                   0100
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  11111111111111111111111111111111  00000000000000000000000000000000                                    11111111111111111111111111111111
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

NOR

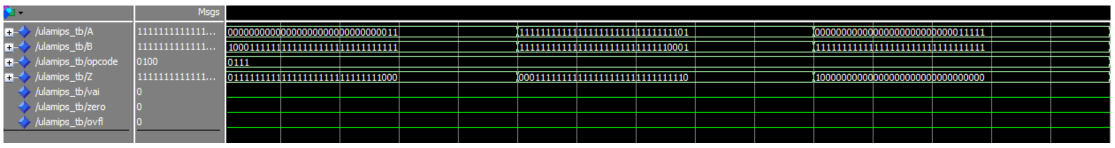

```
/ulamips_tb/A        0000000000000...  11000000000000000000000000000000  00000000000000010101010101010101
/ulamips_tb/B        0111000000000...  11111111111111111111111111111100  11111111111111110110110110110110
/ulamips_tb/opcode   1111              0101
/ulamips_tb/Z        ZZZZZZZZZZZZZ...  00000000000000000000000000000011  00000000000000001000001000001000
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

XOR



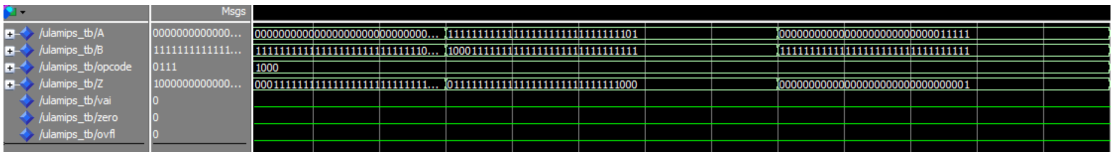
```
                     Msgs
/ulamips_tb/A        1111111111111...   11000000000000000000000000000000   00000000000000001010101010101010 1
/ulamips_tb/B        1111111111111...   11111111111111111111111111111100   11111111111111110110110110110110
/ulamips_tb/opcode   0100               0110
/ulamips_tb/Z        1111111111111...   00111111111111111111111111111100   11111111111111110001110001110001 1
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

SLL



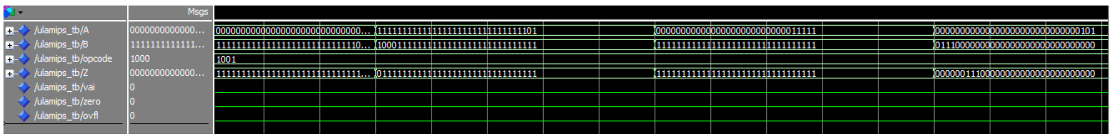
```
                     Msgs
/ulamips_tb/A        1111111111111...   00000000000000000000000000000011   11111111111111111111111111111101   00000000000000000000000000011111
/ulamips_tb/B        1111111111111...   10001111111111111111111111111111   11111111111111111111111111110001   11111111111111111111111111111111
/ulamips_tb/opcode   0100               0111
/ulamips_tb/Z        1111111111111...   01111111111111111111111111111000   00011111111111111111111111111110   10000000000000000000000000000000
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

SRL



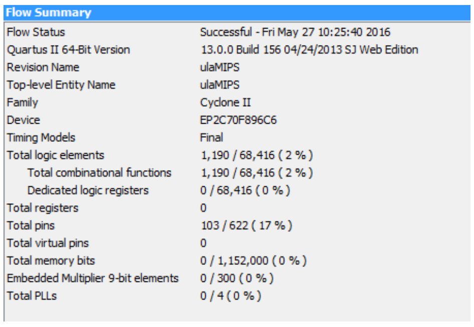
```
                     Msgs
/ulamips_tb/A        0000000000000...   0000000000000000000000000000...   11111111111111111111111111111101   00000000000000000000000000011111
/ulamips_tb/B        1111111111111...   1111111111111111111111111110...   10001111111111111111111111111111   11111111111111111111111111111111
/ulamips_tb/opcode   0111               1000
/ulamips_tb/Z        1000000000000...   0001111111111111111111111111...   01111111111111111111111111111000   00000000000000000000000000000001
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

SRA



```
                     Msgs
/ulamips_tb/A        0000000000000...   0000000000000000000000000000...   11111111111111111111111111111101   00000000000000000000000000011111   00000000000000000000000000000101
/ulamips_tb/B        1111111111111...   1111111111111111111111111110...   10001111111111111111111111111111   11111111111111111111111111111111   01110000000000000000000000000000
/ulamips_tb/opcode   1000               1001
/ulamips_tb/Z        0000000000000...   1111111111111111111111111111...   01111111111111111111111111111111   11111111111111111111111111111111   00000011100000000000000000000000
/ulamips_tb/vai      0
/ulamips_tb/zero     0
/ulamips_tb/ovfl     0
```

Dados de síntese:



Flow Summary	
Flow Status	Successful - Fri May 27 10:25:40 2016
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	ulaMIPS
Top-level Entity Name	ulaMIPS
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	1,190 / 68,416 (2 %)
Total combinational functions	1,190 / 68,416 (2 %)
Dedicated logic registers	0 / 68,416 (0 %)
Total registers	0
Total pins	103 / 622 (17 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

ulaMIPS.vhd:

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

entity ulaMIPS 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 ulaMIPS;
```

```
architecture ulaExec of ulaMIPS is

begin

        process (A, B, opcode)

        constant zeros : std_logic_vector(WSIZE-1 downto 0) := (others => '0');
        constant ones : std_logic_vector(WSIZE-1 downto 0) := (others => '1');

        variable tmp : std_logic_vector(WSIZE-1 downto 0);
        variable soma : std_logic_vector(WSIZE-1 downto 0);
        variable cin, cout : std_logic;

        begin
                vai <= '0';
                ovfl <= '0';
        --  O código da entrada "opcode"determina que tarefa será executada
                case opcode is
                        when "0000"  =>         -- Lógica "E" bit a bit
                                tmp := A and B;
                        when "0001"  =>         -- Lógica "OU" bit a bit
                                tmp := A or B;
                        when "0010"  =>         -- Adição
                                for i in 0 to (WSIZE-1) loop
                                        if(i = 0) then
                                                cin := '0';
                                        else
                                                cin := cout;
                                        end if;
                                        soma(i) := A(i) xor B(i) xor cin;
                                        cout := (A(i) and B(i)) or ((A(i) xor B(i)) and
cin);
                                end loop;
                                vai <= cout;
                                ovfl <= cin xor cout;
                                tmp := soma;
                        when "0011"  =>         -- Subtração
                                for i in 0 to (WSIZE-1) loop
                                        if(i = 0) then
                                                cin := '0';
                                        else
                                                cin := cout;
                                        end if;
                                        soma(i) := A(i) xor B(i) xor cin;
                                        cout := ((not A(i)) and B(i)) or(B(i) and cin)
or(cin and (not A(i)));
                                end loop;
                                vai <= cout;
                                ovfl <= cin xor cout;
```

```
                                        tmp := soma;
                                when "0100"  =>         -- Set on Less Then
                                        if(signed(A) < signed(B)) then
                                                tmp := ones;
                                        else
                                                tmp := zeros;
                                        end if;
                                when "0101"  =>         --  Lógica  "COMPLEMENTO  DE  OU"
bit a bit
                                        tmp := A nor B;
                                when "0110"  =>         -- Lógica "OU ESCLUSIVO" bit a bit
                                        tmp := A xor B;
                                when "0111"  =>         -- Shift Left Lógico
                                        tmp        :=        std_logic_vector(unsigned(B)        sll
to_integer(signed(A)));
                                when "1000"  =>         -- Shift Right Lógico
                                        tmp        :=        std_logic_vector(unsigned(B)        srl
to_integer(signed(A)));
                                when "1001"  =>         -- Shift Right Aritmético
                                        tmp      :=      to_stdlogicvector(to_bitvector(B)      sra
to_integer(signed(A)));
                                when others =>          --   Saída   em   alta   impedância   para
operação indefinida
                                        tmp := (others => 'Z');
                        end case;

                        if(tmp = zeros) then
                                zero <= '1';
                        else
                                zero <= '0';
                        end if;

                        Z <= tmp;

  end process;

end ulaExec;
```

ulaMIPS_tb.vhd:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ulaMIPS_tb IS
END ulaMIPS_tb;
ARCHITECTURE ulaMIPS_arch OF ulaMIPS_tb 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 Z : STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL vai : STD_LOGIC;
SIGNAL zero : STD_LOGIC;
SIGNAL ovfl : 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);
	Z : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
	vai : OUT STD_LOGIC;
	zero : OUT STD_LOGIC;
	ovfl : OUT STD_LOGIC
	);
END COMPONENT;
BEGIN
	i1 : ulaMIPS
	PORT MAP (
-- list connections between master ports and signals
	A => A,
	B => B,
	opcode => opcode,
	Z => Z,
	vai => vai,
	zero => zero,
	ovfl => ovfl
	);

	PROCESS
-- variable declarations
	BEGIN
		-- AND
		A <= (31 | 30 => '1', others => '0');
		B <= (0 | 1 => '0', others => '1');
		opcode <= "0000";
		wait for 10 ns;
		A <= (0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 => '1', others => '0');
		B <= (0 | 3 | 6 | 9 | 12 | 15 => '0', others => '1');
		opcode <= "0000";
		wait for 10 ns;

		-- OR
		A <= (31 | 30 => '1', others => '0');
		B <= (0 | 1 => '0', others => '1');
		opcode <= "0001";
		wait for 10 ns;
		A <= (0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 => '1', others => '0');
```

```
B <= (0 | 3 | 6 | 9 | 12 | 15 => '0', others => '1');
opcode <= "0001";
wait for 10 ns;

-- ADD
A <= (others => '1');
B <= (0 => '1', others => '0');
opcode <= "0010";
wait for 10 ns;
A <= (0 => '1', others => '0');
B <= (31 => '0', others => '1');
opcode <= "0010";
wait for 10 ns;
A <= (others => '0');
B <= (others => '1');
opcode <= "0010";
wait for 10 ns;
A <= (1 | 2 | 3 | 4 | 5 => '1', others => '0');
B <= (3 | 4 | 5 | 6 | 7 => '1', others => '0');
opcode <= "0010";
wait for 10 ns;

-- SUB
A <= (0 => '1', others => '0');
B <= (0 | 31 => '1', others => '0');
opcode <= "0011";
wait for 10 ns;
A <= (others => '1');
B <= (0 => '1', others => '0');
opcode <= "0011";
wait for 10 ns;
A <= (0 => '1', others => '0');
B <= (31 => '0', others => '1');
opcode <= "0011";
wait for 10 ns;
A <= (others => '0');
B <= (others => '1');
opcode <= "0011";
wait for 10 ns;
A <= (0 | 31 => '1', others => '0');
B <= (31 => '0', others => '1');
opcode <= "0011";
wait for 10 ns;

-- SLT
A <= (0 | 1 => '1', others => '0');
B <= (31 | 30 => '0', others => '1');
opcode <= "0100";
wait for 10 ns;
A <= (0 | 1 => '1', others => '0');
```

```
B <= (0 | 1 => '1', others => '0');
opcode <= "0100";
wait for 10 ns;
A <= (others => '0');
B <= (others => '1');
opcode <= "0100";
wait for 10 ns;
A <= (0 | 1 => '0', others => '1');
B <= (others => '1');
opcode <= "0100";
wait for 10 ns;

-- NOR
A <= (31 | 30 => '1', others => '0');
B <= (0 | 1 => '0', others => '1');
opcode <= "0101";
wait for 10 ns;
A <= (0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 => '1', others => '0');
B <= (0 | 3 | 6 | 9 | 12 | 15 => '0', others => '1');
opcode <= "0101";
wait for 10 ns;

-- XOR
A <= (31 | 30 => '1', others => '0');
B <= (0 | 1 => '0', others => '1');
opcode <= "0110";
wait for 10 ns;
A <= (0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 => '1', others => '0');
B <= (0 | 3 | 6 | 9 | 12 | 15 => '0', others => '1');
opcode <= "0110";
wait for 10 ns;

-- SLL
A <= (0 | 1 => '1', others => '0');
B <= (30 | 29 | 28 => '0', others => '1');
opcode <= "0111";
wait for 10 ns;
A <= (1  => '0', others => '1');
B <= (1 | 2 | 3 => '0', others => '1');
opcode <= "0111";
wait for 10 ns;
A <= (0 | 1 | 2 | 3 | 4 => '1', others => '0');
B <= (others => '1');
opcode <= "0111";
wait for 10 ns;

-- SRL
A <= (0 | 1 => '1', others => '0');
B <= (1 | 2 | 3 => '0', others => '1');
opcode <= "1000";
```

```
                wait for 10 ns;
                A <= (1  => '0', others => '1');
                B <= (30 | 29 | 28 => '0', others => '1');
                opcode <= "1000";
                wait for 10 ns;
                A <= (0 | 1 | 2 | 3 | 4 => '1', others => '0');
                B <= (others => '1');
                opcode <= "1000";
                wait for 10 ns;

                -- SRA
                A <= (0 | 1 => '1', others => '0');
                B <= (1 | 2 | 3 => '0', others => '1');
                opcode <= "1001";
                wait for 10 ns;
                A <= (1  => '0', others => '1');
                B <= (30 | 29 | 28 => '0', others => '1');
                opcode <= "1001";
                wait for 10 ns;
                A <= (0 | 1 | 2 | 3 | 4 => '1', others => '0');
                B <= (others => '1');
                opcode <= "1001";
                wait for 10 ns;
                A <= (0 | 2 => '1', others => '0');
                B <= (30 | 29 | 28 => '1', others => '0');
                opcode <= "1001";
                wait for 10 ns;

                opcode <= "1111";
                wait for 10 ns;
                wait;

        END PROCESS;
END ulaMIPS_arch;
```